RICE UNIVERSITY

Detecting Events from Twitter in Real-Time

by

Siqi Zhao

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE

Lin Zhong, Chair
Associate Professor,
Electrical and Computer Engineering

Ashutosh Sabharwal
Professor,
Electrical and Computer Engineering

Devika Subramanian
Professor,
Computer Science

Venu Vasudevan
Adjunct Professor,
Electrical and Computer Engineering

HOUSTON, TEXAS
April 2013

ABSTRACT

Detecting Events from Twitter in Real-Time

by

Siqi Zhao

Twitter is one of the most popular online social networking sites. It provides a unique and novel venue of publishing: it has over 500 million active users around the globe; tweets are brief, limited to 140 characters, an ideal way for people to publish spontaneously. As a result, Twitter has the short delays in reflecting what its users perceive, compared to other venues such as blogs and product reviews.

We design and implement SportSense, which exploits Twitter users as human sensors of the physical world to detect major events in real-time. Using the National Football League (NFL) games as a targeted domain, we report in-depth studies of the delay and trend of tweets, and their dependence on other properties. We present event detection method based on these findings, and demonstrate that it can effectively and accurately extract major game events using open access Twitter data. SportSense has been evolving during the 2010-11, 2011-12 and 2012-13 NFL seasons and it has been collecting hundreds of millions tweets. We provide SportSense API for developers to use our system to create Twitter-enabled applications.

Acknowledgments

I would like to firstly thank my advisor, Professor Lin Zhong, who has guided me throughout my academic life at Rice University and made significant impact on my way to become a researcher.

Dr. Venu Vasudevan deserves my special thanks for his insightful and inspiring advices to me. I also appreciate the help from Professor Devika Subramanian and Professor Ashutosh Sabharwal. Their comments and feedback to this work are of great value.

Finally for all my group members and friends, as well as my parents, I cannot imagine how I can succeed in my research and life without them.

Contents

List of Figures

Chapter 1

Introduction

Twitter has over 500 million active users, who can be can be collectively regarded as human sensors that provide quick, brief output for whatever motivates them to tweet. With the huge amount of data and various users, there are numerous ways to study Twitter. Some investigated the Twitter network properties such as the geographical distribution and information diffusion [1]. Some studied the user properties to measure the user influence and reliability [2]. Some attempted to interpret the output of these human sensors, including efforts that detect major social and physical events such as earthquakes [3], celebrity deaths [4], and presidential elections [5]. However, these works have significant delays and usually perform their processing off-line.

This thesis answers a complementary question*: how good are these human sensors for the real-time detection of less significant but more frequent events, such as those happening in a nationally televised sports game?* The insights gained from answering this question will fuel novel applications that leverage the human sensors in

real-time. The solution can also guide other applications such as reporting traffic accidents.

To answer the above question, we use National Football League (NFL) game events as a case study and build a working system, SportSense. SportSense analyzes tweets collected during NFL games. It infers how excited game watchers are and detects big plays, such as scoring events and turnovers, seconds after they happen.

To achieve our goal of real-time analysis for event detection, we shall retrieve as many relevant tweets as fast as possible and from as many users as possible. In Chapter 3, we summarized the Twitter Application Program Interfaces (API) and their limits. We also discovered undocumented limits in our study. Based on these studies, we deployed the scoping method that can extract enough event related tweets while not exceeding the limits.

Delay or latency is a major consideration for any sensing apparatus. In Chapter 4, we investigate it for the sake of using Twitter as a real-time reading of the human sensors. Many applications require game events and game popularity information to be detected in real-time. Otherwise, the information is meaningless when people are no longer interested in the event. For example, the EPG needs the program information in real-time, advertising campaign is operated near real-time. Therefore, we analyze the delay of the Twitter-based solutions.

We first opt to build SportSense using lexicon and threshold based methods in Chapter 5, which demonstrates the feasibility of utilizing Twitter for real-time event detection [6]. Specifically, we track the post rate of event keywords to maintain 90% of

accuracy and we adopt adaptive sliding window approach to achieve real-time detection speed of shorter than 40 seconds from events take place to events recognized by our system, which is nearly 50% shorter than ESPN website update.

In Chapter 6, we describe a matched filter based event detection method [7]. In this filter-based method, (*i*) we studied the key sensing properties of human sensors, i.e., *delay* and *post rate*, and their dependency on other properties such as device being used, user activeness, and the length of tweets. We find that tweets from different user categories had different properties in terms of delay and post rate. (*ii*) These properties guide the detection algorithms that employ increase rate, threshold as well as the matched filter detector that forms different event templates for fundamental categories of tweets.

In order to further reduce the latency and improve the accuracy, we explore the event related features and leverage statistics method to select the best features in real-time event detection in Chapter 7. In particular, we extracted features regarding to three main categories, i.e. sliding window, tweet characteristics, and user characteristics. Based on these features, we deploy regression based event detection method, which outperforms our previous threshold and filter based method. More importantly, studying these features enables the new or non-predefined events discovery in future work because these features are common in many events.

In addition to algorithm design and improvement, SportSense web application is the first of its kind that demonstrates the feasibility and performance in real-time events detection. The SportSense web application has been continuously operational for the past three NFL seasons, providing game information to NFL fans through web browser as

well as its API, nearly 50% faster than EPSN website updates. While SportSense was built for and tested through NFL games, its techniques can be readily applied to other sports games and even TV shows that have a similarly sized fan population and similar frequencies of major events, e.g., soccer, baseball, and reality shows. The keyword-based tweet retrieval method adopted by SportSense allows it to be easily revised for detecting other events of the physical world by updating the keywords. As a demonstration, we build a Twitter-powered electronic program guide (EPG) for live broadcast TV programs based on SportSense. Therefore, Chapter 8 explains the performance evaluation and web application realization.

Overall, SportSense successfully demonstrates that it is feasible to use Twitter users as human sensors to infer about the physical world in real-time. Our experience makes the following contributions.

1. We show that threshold, filter and regression based methods can be used to build efficient, effective and flexible systems to infer the physical world using Twitter.

2. We demonstrate that the limitation posed by Twitter on its free, public APIs can be effectively overcome. Our work also revealed an undocumented limit.

3. We present the first publicly reported study of Twitter's delay in reflecting the physical world. We show that Twitter is subject to an aggregated delay of ~20 seconds primarily due to human response time.

4. We describe the web service realization of SportSense and a Twitter-powered EPG built on top of SportSense that allows a TV watcher to select a TV program based on peer users' feedback in real-time.

The thesis is organized as follows. In Chapter 2 we provide background on Twitter and survey the related work. In Chapter 3 we introduce Twitter APIs and our data collection method. In Chapter 4 we present the delay characterization that includes delays introduced by the human user, Twitter and analysis. We describe the threshold-based event detection method in Chapter 5, the filter-based event detection method in Chapter 6, and the regression-based event detection method in Chapter 7. We present our system design and examine the event detection performance in Chapter 8. We described our future work in Chapter 9. Then we conclude in Chapter 10.

Chapter 2

Background and Related Work

In this chapter, we provide the background of online social network and survey the prior literature on Twitter analysis. The background of online social network includes the popularity of online social network and the microblogging service, Twitter. The prior literature includes three areas, social network study, user property and influence, event and sentiment detection.

2.1. Online Social Network Background

Online social network refers to the platforms that users can build relationships and share interests with others. Popular social networking sites include MySpace, Facebook, Twitter and Google+, etc. In recent years, social networking sites play more important roles in people's life. In 2005, MySpace attracts more page views than Google. In 2009, Facebook overtook MySpace and became the largest social network site with over 1 billion visits per month [8]. In 2011, 47% of American adults use a social network [9].

Therefore, social network sites become important sources to study the user behavior and relationship.

The emerging trend in social network sites is “real-time” and “location-based”. Twitter becomes the most popular microblog where users can broadcast to the world what they are doing, or what is on their mind within a 140 character limit. When it was emerging we realized that Twitter can potentially become the source to detect what is happening in the world. We propose to leverage Twitter users as human sensors to detect breaking news and events. We now examine the literature related to social network analysis and event detection.

2.2. Related Work

Prior literature is comprised of four major categories. The first category is the topological and content properties. The second is the user property and influence. The third category refers to the event and sentiment detection. The fourth is event detection is video and document. To the best of our knowledge, the measurements of human-Twitter sensor’s delay and its dependency on other properties are novel.

2.2.1. Twitter Network and Content Study

Microblog, as a union of all sensors, firstly attracted attentions from researchers to explore its topological properties. For example, Java *et al.* [1] studied the topological and geographical properties of Twitter. Huang *et al.* [10] analyzed the tagging patterns in Twitter. Kwak *et al.* [11] investigated the entire Twitter atmosphere and information diffusion. Yang *et al*. [12] studied the temporal patterns associated with online content.

Their task is fundamentally different from ours. They focus on the properties of Twitter as a community and information sharing platform.

On the other side, many researchers studied the Twitter content in order to discover what content are valued by users and how to direct user attention to desirable information. Andre *et al*. [13] analyzed the Twitter content from the reader's point of view. They examined what content do Twitter users value and why are some tweets valued more than others. Chen *et al*. [14] studied the content recommendation on Twitter.

2.2.2. Twitter User Property and Influence

To measure the sensor's properties, efforts have been made to classify different kinds of users. Chu *et al*. [15] focused on the classification of human, bot and cyborg accounts on Twitter. Naaman *et al*. [16] divided users into informers, who share information, and meformers, who talk about their own. Weng *et al*. [17] proposed TwitterRank, which utilized the followers/friends information as well as PageRank algorithm to measure the user influence. Cha *et al*. [2] measured the user influence based on indegree, retweets and mentions. Canini *et al*. [18] combined the analysis of the link structure of social networks with topic models of the content to decide whose updates to subscribe in order to maximize the relevance, credibility and quality of the information received. Pal and Counts [19] attempted to find the most interesting and authoritative authors for any given topic based on multiple features of authors including original tweet, conversation tweet, mentions etc. Hecht *et al*. [20] studied the user behavior with regard to the location field in Twitter user profiles. They attempted to use machine learning

techniques to examine how much location information does the average Twitter user discloses implicitly by tweeting.

These works focus on the user's connections and tweet content. Our contribution lies in discovering the correlation between delay and other attributes such as device and user activeness.

2.2.3. Twitter Event and Sentiment Detection

Detecting topics, events and sentiments are closely related to our work. Many treat tweets as text documents and apply text mining techniques to extract topics, classify and cluster tweets, e.g., TwitterStand [4], Tweet the debates [5], and Topical clustering of tweets [21]. Some have extracted information about social and physical events using Twitter or similar services. Sakaki *et al*. [3] and Qu *et al*. [22] investigated the earthquakes detection. Vieweg *et al*. [23] studied grassfire and floods. Hannon *et al*. [24] used post rate of tweets to produce video highlights of the World Cup off-line. Chakrabarti and Punera [25] sought to describe a sport event using Hidden Markov Models trained with tweets collected from the past. Golder *et al*. [26] and Bollen *et al*. [27] detected the national mood and emotion rhythms using Twitter data. In addition, sentiments analysis is another hot topic and is widely applicable to finance and marketing projects [5, 28-31].

There are particularly relevant works, Weng and Lee [32] proposed the event detection algorithm EDCoW (Event Detection with Clustering of Wavelet-based Signals). Becker *et al.* [33] explored to identify real-world event content on Twitter. TwitInfo [30], presented a system for visualizing and summarizing events on Twitter.

None of the above can detect a targeted event in seconds. For example, the authors of TwitInfo developed a streaming algorithm that could discover events using minute-long window that could detect event in real-time but would introduce the delay in the magnitude of minutes. The authors of [3] were able to detect an earthquake from Twitter only hours after it happened despite the "real-time" claim in the paper title. In contrast, we are interested in detecting events in seconds; and SportSense is able to reliably detect a touchdown a few tens of seconds, even before ESPN updates the score on the web page. Orthogonal to our vision of using Twitter users as human sensors, recent proposals of human computation or crowdsourcing use human, in particular Amazon Mechanic Turk, as actuators to perform a given task [34-37]. All these proposals require a participating human to purposefully perform a task for the intended computation, usually with certain incentive. In contrast, we exploit what people are already doing without any usability or financial overhead.

2.2.4. Other Event Detection Research

Event detection in time series has been widely studied. Many investigated the subsequence matching for given query sequence and longer time series. Faloutsos *et al.* [38] presented an indexing method to locate 1-dimensional subsequences within a collection of sequences. Agrawal *et al.* [39] proposed an indexing method for time sequences for processing similarity queries. Moon *et al.* [40, 41] proposed subsequence matching method with the methods of constructing windows and utilizing sliding windows. On the other hand, many attempted to discover interesting patterns, such as frequently appearing or abnormal patterns. Chiu *et al.* [42] proposed the algorithm that can find approximately repeated subsequences. Keogh *et al.* [43] introduced techniques

that define a pattern surprising if its occurrence differs substantially from that expected by chance. Keogh *et al.* [44] investigated finding time series discords, which are subsequences of a longer time series that are maximally different from all the rest of time series subsequences. Chan and Mahoney [45] provided algorithms that produce anomaly scores in an online manner for real life monitoring tasks.

Event detection for sports games has been studied by the video analysis research community by using computation instead of human as sensors, e.g. [46-48]. Multimodal approaches have been studied for video summarization in online presentation [49] and sports games [50, 51]. Petridis *et al*. [52] and Xu *et al* [22] used MPEG-7 and webcast text to extract sports events. Unfortunately, the video content and text information leveraged by the above work is not always available, especially in real-time. For example, NFL games include the closed captioning text in the video but do not offer webcast text because of the copy right issue [2].

There exists extensive prior literature about topic and event detection in document domain. Wei and Lee [53] investigated the use and extension of text retrieval and clustering to automatically detect novel events from a temporally-ordered stream of news stories. Similarly, Yang *et al*. [54] proposed an approach to detect new event in temporally –sequenced streams of documents. Fung *et al*. [55] attempted to detect a set of features for breaking events. Kumaran and Allan [56] utilized text classification techniques and named entities to improve the new event detection performance. Li *et al*. [57] proposed the algorithm that modeled both content and time information to discover previously unidentified events in historical news corpus. These works are focused formal documents such as news articles, which are essentially different from Twitter content.

Since tweets are limited to 140 characters, they are short, unstructured and informal so that these methods cannot maintain the same topic and event detection performance on tweets. Petrovic *et al* [58] attempted to address the problem of detecting new events from a stream of Twitter posts. Their system works in the streaming model and takes constant time to process each document. Their system is good at detecting the general and high level topics in a day with hours of delay. Our work is unique in that we study Twitter as the real-time output of the human sensors to infer the physical world. We investigate the delay properties of users during targeted events. And we utilize the user's property to detect events in real-time.

Chapter 3

Data Collection and Event Tagging

During American football games, Twitter has a high volume of tweets from football fans, tweeting about football events that they find exciting or notable. SportSense leverages this activity, associating particular streams of tweets with game events (e.g. Touchdowns, Field Goals) to perform robust event detection. In this chapter, we introduce the data collection technique that includes Twitter APIs, targeted games, and lexicon-based game tweets separation.

3.1. Twitter APIs

To achieve our goal of real-time analysis for event recognition, we retrieve as many relevant tweets as fast as possible and from as many users as possible. Twitter provides three application programming interfaces (APIs), i.e., REST, Search and Streaming API. Figure 1 shows the database structure and the APIs. In the figure, the

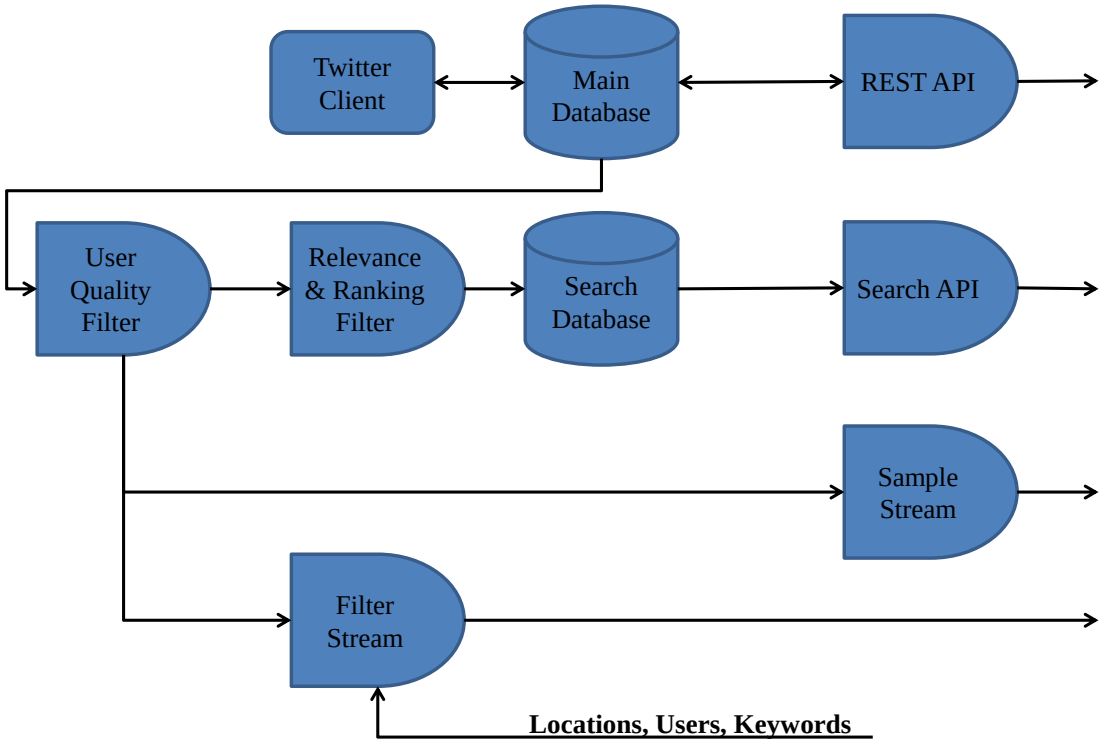


Figure 1 Twitter APIs and database structure. The top arrow on the right represents REST API, the second arrow represents Search API, the bottom two represent the Streaming API.

arrows on the right represent APIs. The top arrow represents the REST API. The second one represents the Search API. The bottom two arrows represent the Streaming API.

The Representational State Transfer (REST) API allows developers to access core Twitter data stored in the main database, which contains all the tweets. Through the REST API, developers can retrieve Twitter data including user information and chronological tweets. For example, the *home timeline* includes the 20 most recent tweets on a user's home page; the *public timeline* returns the 20 most recent tweets in every 60 seconds. These limitations make the REST API not particularly suitable for real-time tweet collection; the REST API is best for collecting a large number of tweets from specific user IDs off-line. In our study, we used it to collect tweets from NFL followers posted during the game for the 2010 Super Bowl off-line.

The *Search API* will return tweets that match a specified query; however it will only search a limited subset of tweets posted in past 7 days in the main database. The query parameters include time, location, language etc. Twitter limits the return results to 100 tweets per request. Although the Search API is able to collect tweets in real-time, one cannot control the topic of the returned tweets. Twitter limits the request rate to the REST and Search API to 150 per hour by default. It previously allowed up to 20,000 search requests per hour from white-listed IPs (about six requests per second), but, unfortunately, Twitter no longer grants white-listing requests since Feb 2011 [59]. This limitation makes the REST and Search APIs unsuitable for real-time event detection.

The *Streaming* API [60] offers near real-time access to Tweets in *sampled* and *filtered* forms. The *filtered* method returns public tweets that match one or more filter predicates, including *follow*, *track*, and *location*, corresponding to user ID, keyword and location, respectively. Twitter applies a User Quality Filter to remove low quality tweets such as spams from the Streaming API. The quality of service of the Streaming API is best-effort, unordered and generally at-least-once; the latency from tweet creation to delivery on the API is usually within one second [61]. However, reasonably focused track and location predicates will return all occurrences in the full stream of public statuses. Overly broad predicates will cause the output to be periodically limited [10]. Our experience shows that the Streaming API is better than either the REST or Search API for real-time game event recognition for three reasons: all the tweets returned are up to date; there is no rate limit; and the track filter predicate allows us to collect tweets that are related to the game of interest using keywords. Although there is no explicit rate limit, we

cannot obtain all public tweets and we will report our observation of an undocumented restriction in the Streaming API.

3.2. Targeted Games

We use live broadcast the US National Football League (NFL) games as benchmarks. We collected tweets from the 2010 Super Bowl, over 200 games in the 2010-11, 2011-12 and 2012-13 seasons including the playoffs and Super Bowl. First, for the 2010 Super Bowl, we collected the tweets posted during the game, by followers of the NFL twitter account, or simply *NFL followers*, using the REST API. Overall, we collected over half a million tweets from 45,000 NFL followers. Although these tweets were collected off-line, they helped us gain insights into the keywords for collecting tweets in real-time with the Streaming API.

For the 2010-11, 2011-12 and 2012-13 seasons NFL games, we collected tweets during game time using the Streaming API and game keywords identified from the 2010 Super Bowl. We collected the tweets and their metadata such as tweet source, created time, location, and device. These tweets were analyzed for event recognition in real-time through a web service described below. For the regular season games and playoffs, we collected more than 300 million game-related tweets including 200+ games, from 200 million users. In the most tweeted game, we collected about 1 million game-related tweets from over half a million users in 2011 Super Bowl. The evaluation of our solutions was performed in real-time when a game was ongoing and was repeated with trace-based emulation off-line if necessary.

3.3. Game Tweets Separation

We next provide the rationale behind the keywords used to retrieve game-related tweets for real-time analysis. Here we use the tweets retrieved with the REST API from NFL followers posted during the 2010 Super Bowl. When performing real-time analysis of tweets for a game of interest, it is very important to focus on tweets that are actually talking about the game, not only because tweets unrelated to the game will interfere with the analysis but also because Twitter limits the rate tweets can be retrieved.

3.3.1. Lexicon-based Game Tweets Separation

We find that such keywords include game terminology and team names. To examine the relationship between the game and tweets posted during the game, we compute and rank the term frequencies of all words that appeared in the tweets posted during the game. After running a stemming algorithm to eliminate misspelling words, we find that the top 10 most frequent words are either game terminology or team names.

Are these keywords sufficient to extract game-related tweets? To answer this question, we randomly select 5% of the tweets, about 2,000, posted during the game by the NFL followers. We manually examined all of these tweets to determine if each of them was related to the game. Half of these tweets had at least one of the top 10 keywords. There were some tweets with incomplete sentences; and we treated the uncertain ones as unrelated. Using the manually classified set of tweets as the ground truth, we find that extraction by the top 10 keywords is surprisingly effective, achieving a false negative rate below 9% and a false positive rate below 5%.

Further, we found that the team names appear in over 60% of the game-related tweets. Therefore, we rely on the team names to collect data when multiple games take place at the same time and attribute these tweets to games based on the mentioned team names.

The performance of the lexicon-based heuristic can be further improved by examining the falsely identified tweets. The first major source of error is the foreign languages tweets using the keywords or Twitter hashtags. If these tweets are not considered, the false negative and false positive rates will be reduced to 5.2% and 2.8%, respectively. The second major source of error is misspelling because Twitter users can spell words wrong either deliberately or by mistake. By applying the spelling check engine and regular expression applications, we can reduce the false positive and false negative rate to 4% and 2%.

3.3.2. Undocumented Twitter API Limit

While our original system (which will be introduced later) performed consistently well for all the games in the 2010-11 season games, it catastrophically failed to recognize any event during the 2011 Super Bowl. In this sub-section, we present results from our investigation of the failure and provide enhanced solutions to cope with similar situations.

Since we recorded all the tweets for the games, we were able to investigate what happened. Our investigation revealed that the failure was caused by an implicit throughput limit by Twitter for its Streaming API: about 50 tweets per second. Twitter API documentation only explicitly states two other limits: first, “reasonably focused track

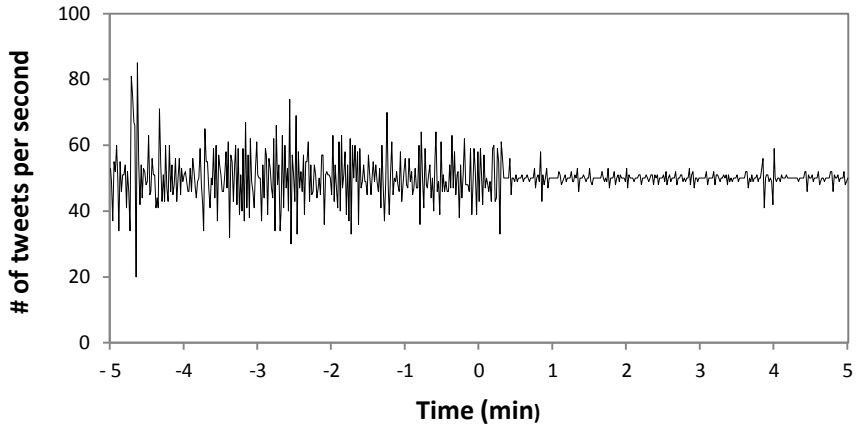


Figure 2: The second-wise post rate of collected tweets for five minutes before and after the first touchdown in the 2011 Super Bowl

and location predicates will return all occurrences in the full Firehose stream of public statuses" and "broad predicates will produce limited streams that will tend to be a subset of the statuses/sample streams" (1% sampling rate).

We discovered the undocumented throughout limit of the Streaming API by analyzing the tweets collected during the Super bowl. We notice that the post rate does not increase considerably as usual when events happen. We therefore zoom in to examine the number of tweets collected in each second from 5 minute before to 5 minutes after the event. To our surprise, we find that the throughput is floating around 50 tweets per second and does not increase when an event happens. In particular, the average number of tweets collected per second is 50.1 and the limit becomes stricter immediately after the event because the standard deviation reduces dramatically from 8.4 to 1.9, as shown in Figure 2. Had not been this limit, we should have collected tweets at much higher rate when an event happens because, according to Twitter [62], the overall post rate doubled during major Super Bowl events.

The original event detection method failed to detect any rises in the post rate of tweets collected from the Streaming API because the true post rate during the Super Bowl was almost always very high, keeping our data collection at the throughput limit of 50 tweets per second. In contrast, in the regular season and playoffs, the rate limit was never reached so that the event detection stage did see the post rate rises due to events.

Chapter 4

Delay Characterization

Delay or latency is an important consideration for any sensing apparatus. In this chapter, we investigate it for the sake of using Twitter as a real-time reading of the human sensors. Many applications require audience sentiments and game events be detected in real-time. The information provided to end users or advertisers must be on time. Otherwise, the information is meaningless when people are no longer interested in the event. For example, the EPG needs the game information in real time, advertising campaign is operated near real time. Therefore, we analyze the delay of our Twitter-based solutions in this section.

Three sources contribute to the delay of our Twitter-based solution. The delay introduced by Twitter users is the period between people perceiving a social or physical event and posting the tweet. This *human delay* is mainly determined by how fast twitter users perceive the event, how fast they react to it, and how fast they type the tweet. Twitter itself also introduces delay in providing tweets through its API; *the Twitter delay*

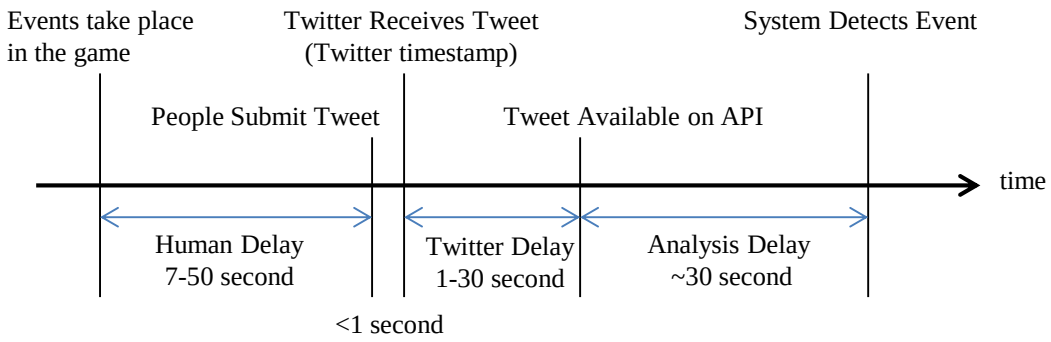


Figure 3: Important timestamps to study the delays

may be affected by Twitter workload, the user quality filter process or the Twitter's index mechanism. Finally, our data collection and analysis also introduce a delay or *analysis delay*.

To study these delays, we must be able to know the following important time stamps: when an event takes place, when a related tweet is submitted to Twitter by user, when the tweet is retrieved through the Stream API, and when the event is recognized. Figure 3 summarizes the important timestamps in studying delay. While the last two time stamps are trivial to obtain, the first two requires some extra work. First, oddly all media, e.g., ESPN channel, NFL website and sports newspapers, record the game events in the *game time*, the time corresponding to the game progress. Moreover, the NFL game rewind is commercial free, making the time a few minutes apart from the absolute time. Instead, we observed the broadcasts of games, spanning the 2011 and 2012 Super Bowl, some 2011-12 regular season games, which include more than 100 touchdowns. We monitored regular season games at different times and cities, making our sample as random as possible. We manually marked the real clock time of game events. Due to copyright issues of rebroadcasting games, we were not able to crowdsource this task.

Secondly, Twitter time stamps a tweet when it is received. We call this time the *Twitter time*. To accurately estimate the human delay, we must know if there is a significant delay between when a tweet is submitted and when it is time-stamped by Twitter. We found the delay is negligible, i.e., one second or shorter, which will be proved by experiment in section 4.2.

4.1. Human Delay

We estimate the human delay as the time difference between when a game event happens and when Twitter time stamps a tweet talking about the event. We found that the human delay is as short as 7 seconds and people using mobile devices tweet are even slower than people using non-mobile devices.

We extract the tweets posted after the event and contain the event keyword. Then we read the tweets to select the tweet that is the first discussing the event in a narrative statement rather than prediction or anticipation. People predict events, discuss about the past events that introduce some noise but in a low post rate and a low frequency, i.e. at most 2 in a second and usually null. Since two events of the same type do not usually happen in a short interval (several minutes), we assume the tweets that mentioned a happened event posted several minutes after an event of the same type are discussing about the event that just happened. Take the 2011 Super Bowl as an example, our results, as summarized by Figure 4, show that the average human delay is 17 seconds. The shortest delay is only 13 seconds and the longest is 27 seconds. Interestingly, touchdowns saw shorter human delays than other less significant ones, indicating Twitter users post faster for more significant events.

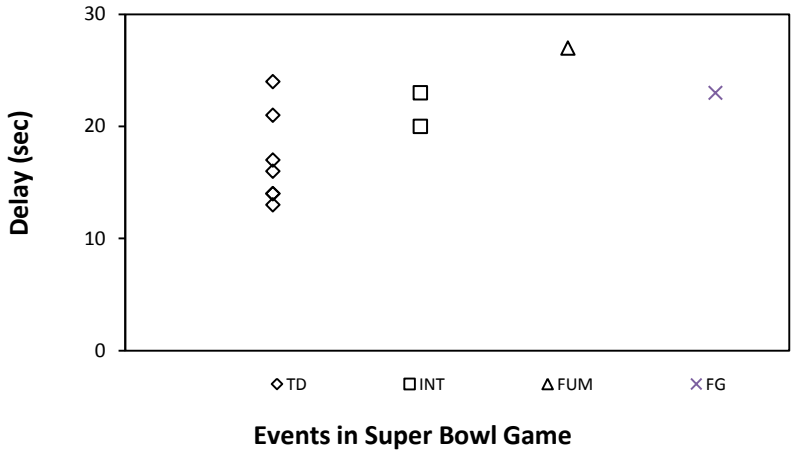


Figure 4: The distribution of human delay in 2011 Super Bowl

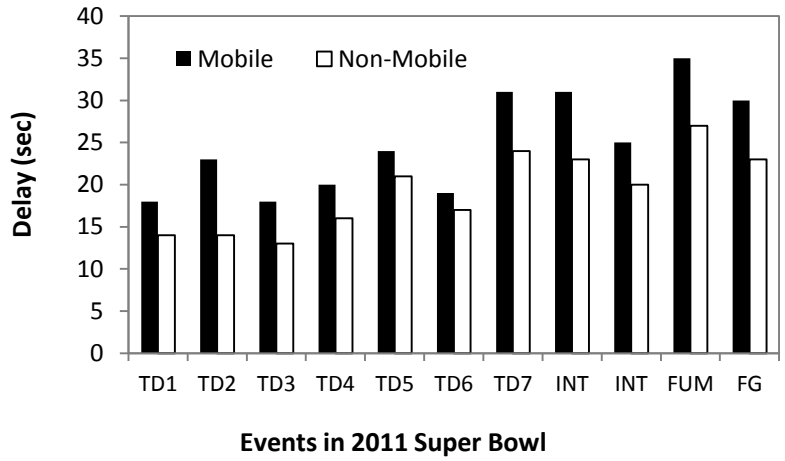


Figure 5: The delay from mobile and non-mobile when events happen.

It must be noted that there is a short delay in broadcasting live materials used to prevent profanity, bloopers, violence, or other undesirable material from making it to air. This delay may vary in different locations and is approximately 7 to 50 seconds. As a result, tweets by game watchers from the stadium should be posted earlier than those from homes. However, we were not able to collect a large number of tweets that can be identified as from the stadium because only a small fraction of Twitter users allow their location to be included in their tweets.

We had expected tweets from mobile devices to have a shorter delay because the effort to switch from game watching to tweeting seems to be lower on mobile devices than on PC or laptops. Our results show that there is no significant difference in regular season games. However, in the Super Bowl, results show that non-mobile tweets are faster. By examining the source applications, we observed that nearly 40% of game-related tweets were from Twitter clients on recognizable mobile devices, i.e. iPhone, BlackBerry, Android, txt, mobile, HTC, MOTO, and iPad. The actual percentage of tweets from mobile devices should be more than 40% because there are tweets posted from the clients that are both available to mobile and non-mobile devices. In this study, we only consider the tweets from the recognizable mobile devices.

In the Super Bowl, we inspect 7 touchdowns, 2 interceptions, 1 fumble, and 1 field goal. The results illustrate that the non-mobile users react 3 to 5 seconds faster in all the 11 events, as shown in Figure 5. The number of tweets posted from non-mobile devices increases faster as well. Take the 1st touchdown for example, the posting rate from non-mobile devices increases from null to the maximum (27 per second) in 10 seconds whereas that from mobile devices starts 3 second later and takes 36 seconds to reach the maximum (23 per seconds).

The possible reasons that cause the results include typing speed and device/network delay. Although mobile devices, mainly smartphones, are more portable and convenient, the typing speed on mobile devices is still lower than that on PC or laptops. On the other hand, the mobile devices may suffer from the network condition [63]. Consequently, mobile users spend more time in the *human delay*.

4.2. Twitter Delay

Twitter Streaming API allows keyword-based retrieval in real time. We are interested in how much delay Twitter introduced to its Streaming API since there is no published study regarding it. We calculate the Twitter delay as the difference between the Twitter timestamp of a tweet and the time we obtain the same tweet from Twitter.

We make two observations regarding the Twitter delay. The delay is about 30 seconds for tweets only contain custom, random keywords and about one second for tweets with Twitter promoted, widely used keywords, e.g., Twitter promoted #sb45 during the 2011 Super Bowl. Second, the delay is fairly independent of the post rate.

Our observations are based on two measurement studies. The first measurement is similar to that used to measure the human delay. Tweets that only contain the system time and the keyword "Twittertime" are posted and retrieved through the Streaming API to calculate the Twitter delay. 10 tweets were randomly posted in 2 hours, once in a weekday afternoon without special event, and the other during the NCAA Men's Basketball Final. The Twitter delay is consistently about 30 seconds.

The second measurement was performed during the NCAA Men's Basketball Final with a keyword promoted by Twitter, *#MM2011* meaning March Madness 2011, and the widely used team names, Butler and UConn, to collect data. Like the NFL playoffs, the posting rate was high during the game (20 per second). We found the Twitter delay is consistent about one second, much shorter than the 30 seconds we observed for retrieval with custom keywords. Even the team names, which are widely

used but not promoted by Twitter, when used as the keyword, return tweets in about one second, which is consistent with Twitter's own claim [61].

It is surprising to see the choice of keywords makes such a huge difference in Twitter delay: 30 seconds vs. 1 second. Without information from inside Twitter, we have to speculate the reason. Twitter maintains its Streaming API result quality by applying the status quality metrics, in some instances, in combination with other metrics, to filter the tweets [10]. The goal is to eliminate spams, inappropriate, or repetitious tweets. Therefore, the tweets contains rarely used keywords may take longer to pass the process. In addition, developers in the Twitter development talk, an online forum in Google groups [64], mentioned that Twitter may index interesting tweets to improve the search speed. Since the Streaming API with filter predicates, i.e., follow, track, or locations, involves search, we speculate such indexing helps significantly reduce the Twitter delay for Twitter promoted keywords and popular keywords such as team names[54, 65].

We also observe that the post rate does not impact the Twitter delay noticeably by comparing the delay during both low and high traffic time, i.e., during the March Madness 2011.

4.3. Analysis Delay

The data collection and processing to recognize events also introduce delay. We have implemented our solutions in a way to maximize the parallelism of data analysis as

will be described in the next section. Here we report the measurement of the delay in our implementation.

We estimate that the analysis is less than 20 seconds on average. In the threshold-based detection method to be described in Chapter 5, the delay introduced by the detection stage is bounded by the sliding window size, 10 to 60 seconds or a delay of 5-30 seconds. For example, the average delay from the sliding window and the computation is about 20 seconds. Because the human delay is about 20 seconds and the Twitter delay is about one second, our solution recognizes a game event around 40 seconds on average after the event takes place. The sliding window used in event detection contributes a lot to the overall delay. Interestingly, the higher the post rate, the shorter window will be used and, therefore, introduce a shorter delay. In later chapters we will show that for major events, such as touchdowns, the overall delays can be as short as 30 seconds. To appreciate the delay achieved by our Twitter analysis, we find that the ESPN web page has 90 seconds lag in updating the latest score changes.

Chapter 5

Threshold-Based Event Detection

In this section, we first describe the solution for event detection and then present techniques to improve its performance in accuracy and responsiveness. We will describe our solution in the context of NFL games and detecting big plays.

5.1. Basic Lexicon-Based Event Detection

The key rationale of event detection is that an interesting event, either in physical world or reported in mass media, may immediately trigger its spectators to talk about it, especially through social media like Twitter in real-time. With a large number of spectators, even if only a small fraction of them talk about the physical world event on Twitter, there will be a noticeable increase in tweets related to the event in real-time. Such increases have been noted for various well-spectated events, including earthquake [66, 67] and celebrity death [68] . Our goal is to detect if an event has happened as fast as possible by analyzing scoped tweets collected from Twitter's free Streaming API.

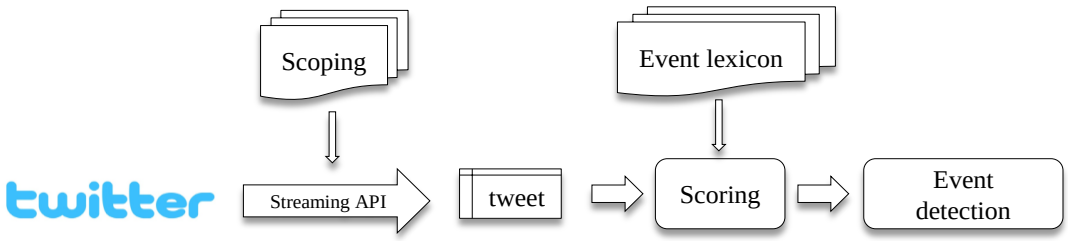


Figure 6: System architecture of SportSense.

The basic two-stage event detection solution is again based on a lexicon defined by the target event, or event lexicon. It operates on the scoped stream retrieved using the lexicon-based method described in Chapter 3. It has two critical steps. First, it employs the event lexicon to score each tweet in the stream between 0 and 1 for how indicative the tweet is about the target event. The scoring method will be described later in this section. Second, it calculates the aggregated score in a time window to declare the event.

Every second, we detect if an event just happened by examining scores of tweets posted in the past 60 seconds fixed time window. To minimize the delay, we adopt an adaptive window approach, which will be explained in next section. By adding the scores of the corresponding tweets, we can calculate a score for the window, and the scores for its first and second halves. We analyze different kinds of events independently to minimize the interference between events. We declare an event just happened if (*i*) the percentage increase of the scores of half windows is over a predefined threshold; and (*ii*) the score of the entire window is also over a predefined threshold. The rationales are simple: the percentage change indicates the trend of an event related discussion; the window must contain enough event-indicative tweets to make a reliable decision. We determine the threshold by examining the events in our off-line dataset of 2010 Super Bowl.

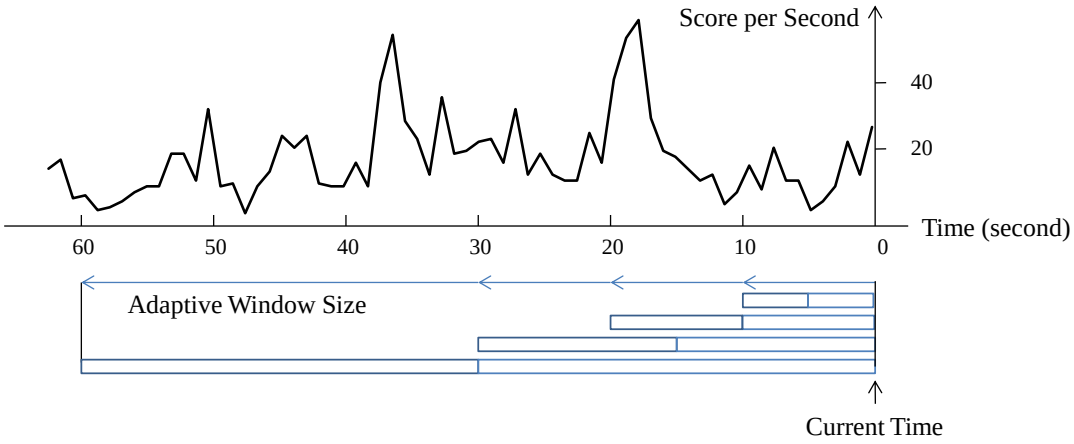


Figure 7: A moving window advancing every second with adaptive size, which increments from 10, 20, 30, to 60 seconds, to examine the aggregated tweets scores in two sub-windows.

We score each tweets using the event lexicon for the simplicity and consistency with scoping. It is simple in implementation and analysis that benefits to the performance in real-time. Comparing to statistical learning approach, lexicon-based approach does not require training data, which is not always available. For example, terrorist attack or celebrity death has no training data available in advance, while the keywords are predictable. On the other hand, we have shown in sub-section 3.3.1 that the scoping lexicon is effective in collecting data and reducing the noises. It is practical to continue using lexicon approach for event detection.

We compose a lexicon for every frequent event in NFL games, i.e., touchdown, interception, fumble, and field goal. Intuitively the event names should be most indicatively if the corresponding event actually took place because a tweet has no more than 140 characters and the event name is indispensable to describe the event. We assign

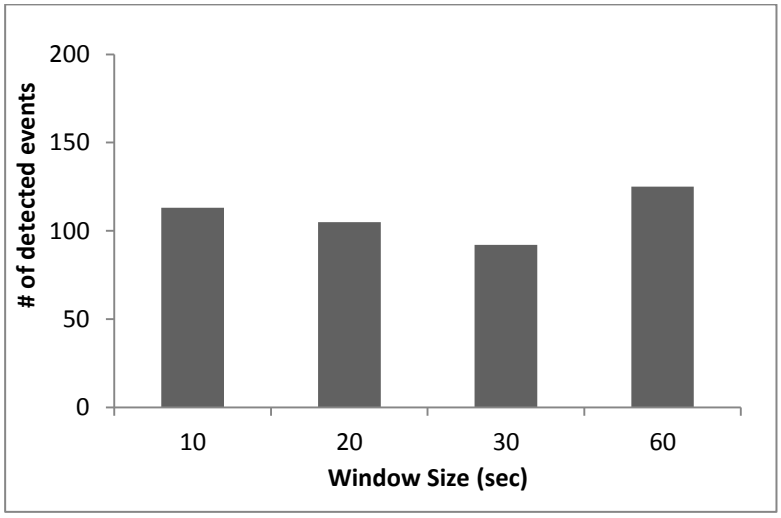


Figure 8: The distribution of events detected with various window sizes

binary scores, 0 and 1, to tweets corresponding to the absence and presence of event names. For different types of events, we count them separately.

Is this lexicon-based approach vulnerable to the undocumented 50 TPS limit? We find that even in the 2011 Super Bowl with the tweet rate saturated to the limit, the post rates of these event keywords can clearly tell when the event happens. For instance, the post rate is almost zero when no event happens while the spike is obvious when event happens.

5.2. Adaptive Sliding Detection Window

To address the real-time detection challenge, we adopt an adaptive window size approach because the window size will have a significant impact on the tradeoff between the delay and accuracy of event detection. A shorter window will lead to a smaller delay but may have a poor performance when the post rate is low and there are not many tweets posted during the window. To achieve the best tradeoff, we devise the window size,

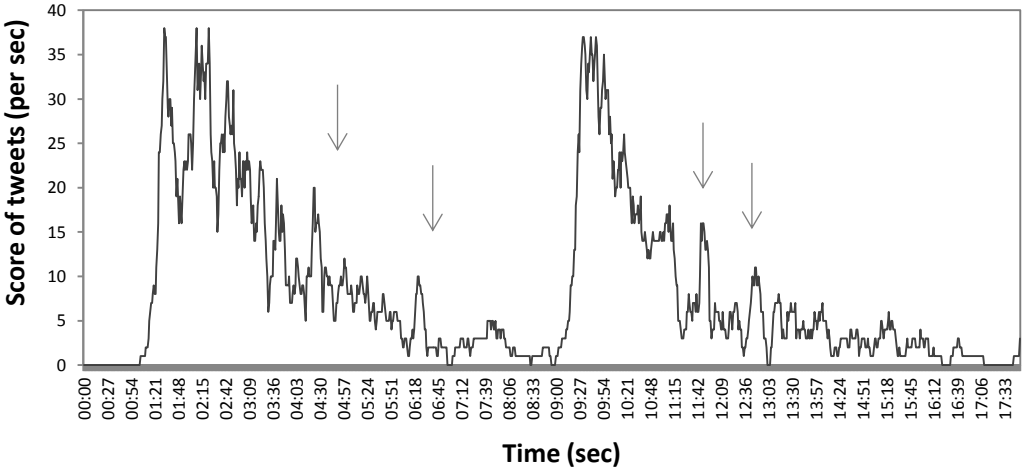


Figure 9: Score of tweets for two consecutive events, small spikes indicated by the arrows will cause duplicate event alerts due to the small detection window size.

adaptive to the post rate within the window, from 10, 20, 30, to 60 seconds as shown in Figure 7. At every second, we will start from the shortest window, 10 seconds, to examine the scores in the window. If the scores passed the threshold, we continue to next processes. Otherwise, we increment the window size until the maximum window size reached or an event declared.

Figure 8 shows the number of events detected in each window size; half of events can be detected using the window size less than 20 seconds and more than 2/3 of events can be detected using the window size less than 30 seconds. Since we halve the window to detect events, the delay of the system is half of the window size. As a result, this method will introduce less than 10 second delay for half of the events.

5.3. Duplicate Event Alerts Prevention

Another challenge we face is the duplicate event alerts that is caused by the intensive discussion among Twitter users. Since an event will trigger a large scale

discussion. Therefore the events detection algorithm may repetitively recognize the event during the discussion. Short window size is the primary reason for this problem. Since fewer tweets are analyzed in a short window, the events detection algorithm has limited knowledge on the score trend. For example, while the general trend is descending, the trend may increase due to the long-lasting discussion as indicated by the arrows in Figure 9. Moreover, since we perform the events detection in real-time, at the start of score increase, we have no clue if this is a small spike in the discussion or a big spike corresponding to another event.

We solve this problem from two aspects. First, we assume that no events of the same kind can happen within five minutes. On the other hand, if any event happens multiple times in a short period, e.g. five minutes, they may not be separable because various delay factors, which will be introduced in next section, will result in the merge of tweets from multiple events. In NFL games, the major events usually lead to the end of this play (another team takes over the possession of ball). It takes time for both teams to substitute players in the new play so that same type of event is unlikely to happen within a short period of time. Second, we pause the adaptive window size approach and adopt the 60 second fixed window size for another five minutes in detecting the event of the same kind so that the algorithm can surpass the small spikes such as those indicated by arrows in Figure 9.

5.4. Weak Event Detection Attempt

Weak events refer to the events that draw less attention and, therefore, lead to fewer tweets and smaller post rate increases. Weak events are vulnerable to noises, such

Events Detection

```
1. At each second, initialize window size as 10 seconds;
2. Score the tweets in the window;
3. Post rate ratio = (score in the first half) / (score in the
   second half of slide window);
4. If (post rate ratio < threshold)
            Increase window size until 60 seconds;

            Go to step 2;

         Else

            Proceed to event recognition;
```

Event Recognition

```
1. Pre-processing
2. Compute the post rate of pre-defined event keywords in the
   second half of the window;
3. If (pre-defined event keyword appears > threshold) and (last
   event of the same type > 5 minute ago)
            Recognize the event;
```

Figure 10: Threshold-based two-stage event detection solution

as discussion or anticipation. For example, field goal is one of the weak events because it is a secondary scoring method that worth only 3 points as opposed to the 6 points touchdown. People do not pay attention to the field goal unless the field goal can change the game result.

We tackle this problem by improving the signal noise ratio in scoring. In particular, we include more positive weighted terms to the event lexicon to strengthen the power of signals. When an event happens, some terms are frequently mentioned in common with the event title, e.g. good, yard, attempt, lead, kick, score, and got. Therefore, we add these terms to the lexicon and assign positive values (e.g. +1) to these terms. On the other hand, we include negative weighted terms to reduce the power of noises. When no event happens, some terms, such as need, miss, and fail, are likely to

appear in common with the event title to express anticipation. We also add these terms to the lexicon but assign negative values (e.g. -1) to them. As a result, the tweet score is calculated based on this revised lexicon.

In summary, we design a two-stage solution in which an event is first detected and then recognized as described by Figure 10. The two-stage structure helps reduce the computational load significantly because detection can be achieved without analyzing the content of tweets. We applied this solution to detect pre-defined football game events including touchdown, interception, fumble, and field goal for nearly a hundred NFL games in real-time.

Chapter 6

Filter-Based Event Detection

In previous chapter, the threshold-based method proves that we can effectively extract useful information from social network in real-time without training data. In this chapter, we take the user and tweet characteristics into consideration. For example, users who post from mobile phones may have different behavior from users who post from computers in terms of speed and accuracy. We first investigate into the user and tweet characteristics after events. Guided by the characteristics, we design a filter-based event detection method based on signal detection technique in order to detect events more accurately and faster.

6.1. Characterizing the Human Response to Events

In this section, we examine the properties of tweets from various users after an event. We analyze the tweet responses of 100 touchdown events across 18 NFL games, investigating characteristics such as the device type, location, user activeness, and the

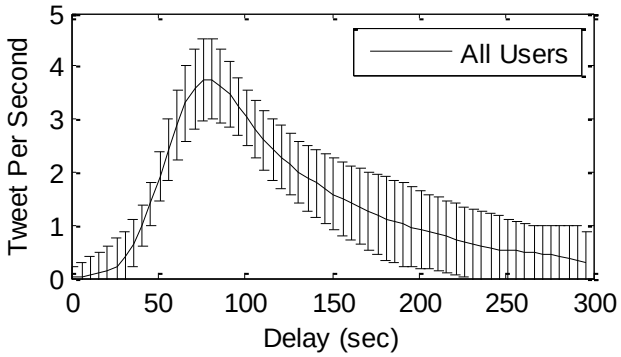


Figure 11: The tweet response among all tweets.

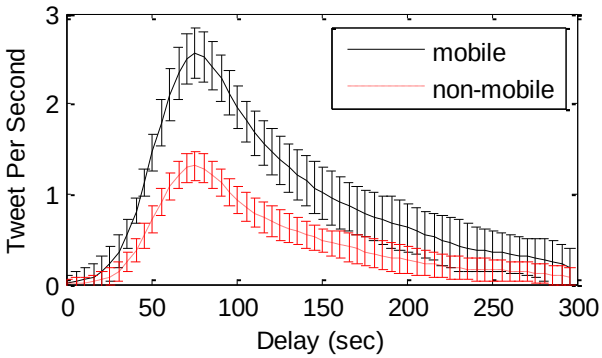


Figure 12: The tweet response from mobile and non-mobile devices.

number of words in a tweet. By modeling the nature of the tweets from these groups, we can later develop a precise event detection method (in Section 6.2). To analyze this, we use a *Human Delay* metric. Numerically, this is the interval between the event occurrence and the tweet's timestamp as explained in Section 4.1. Human delay is influenced by how fast users perceive an event, react to an event, and type a tweet.

We show the tweet response following an event in Figure 11. The x-axis represents the human delay of the tweets, while the y-axis represents the volume of tweets posted per second. We use the error bar to present one standard deviation of uncertainty to represent the variation between the events. From the figure, we can see there are few tweets in the first 50 seconds after an event happens, but the post rates both

increase to a peak at 75 seconds after the event happens. Interestingly, among all the games we observed, the first touchdown-related tweet arrived between 7 seconds and 50 seconds after events.

6.1.1. Mobile Devices Have Higher Post Rate

Many Twitter users post tweets from their mobile devices, e.g., smartphones and tablets. Because of the immediate availability of mobile devices and the significantly different user interfaces, we expect tweets from these different classes of devices to have different delay characteristics.

We consider tweets from Twitter clients on recognizable mobile devices, i.e., iPhone, BlackBerry, Android, HTC, MOTO, as well as mobile browsers and short messages (txt) as introduced in Section 4.1. The portion of tweets from mobile devices increases rapidly in 2011-12 season to over 50%.

We compare mobile and non-mobile responses as shown in Figure 12. The response magnitude on mobile device is twice as large. The peak delays are close; mobile devices reached the peak 2 seconds earlier, but their average delay was longer by 4 seconds. More importantly, we find that the event response on non-mobile devices is more consistent, as evident by the smaller deviation.

Typing speed and device/network delay influence the different delays. The typing speed on mobile devices is still lower than that on PC or laptops. Often, users need to unlock the device and open the Twitter application before tweeting. In addition, mobile devices may suffer from slow network conditions [69]. However, mobile devices are more portable and convenient to use when in front of a TV, at home, in bars, etc.

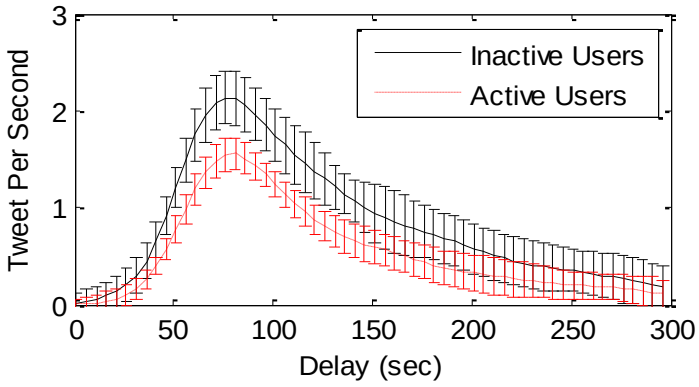


Figure 13: The tweet responses for active and inactive users.

6.1.2. Active Users Post Slower

During NFL games, we find that users have diversified activeness. While most users post a few tweets, e.g., fewer than 5 tweets, in a game, some post dozens of tweets. We define the activeness as how many tweets the user has posted since the beginning of the game. During the game, we track how many tweets a user has posted and compute the average number of tweets per user. At a certain time, if the user has posted more tweets than the average, they are considered active users. Otherwise, they are considered inactive users.

We compare the tweet responses of active and inactive users and illustrate it in Figure 13. For both active and inactive users, the post rates start increasing rapidly from 30 seconds after event happens and reach the peak at the same time. Overall, the average and median delays for active users are about 5 seconds longer than for inactive users. Thus, although active users post more frequently than inactive users, their responses are typically more delayed.

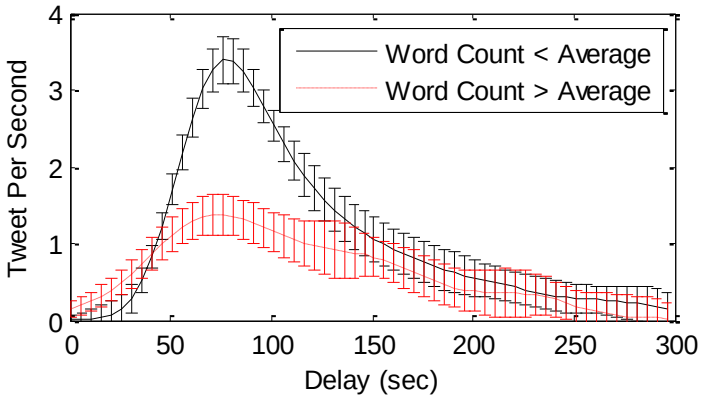


Figure 14: The tweet responses of short and long tweets.

6.1.3. More Short Tweets after an Event

In this section we focus on the word count of tweets after an event happens. We observed that users often post short tweets with exclamation marks and repeated letters to express their excitement. e.g., *New York Jets Touchdownnnnn!!!* Therefore, we expect tweets with few words to correlate with the event. In our data, we found that the tweet word count decreased shortly after an event. In particular, the average word count of tweets decreases from 12 words to 6 words within 60 seconds after an event.

To leverage this result, we categorize tweets into long and short tweets by comparing the tweet word count with the average word count value during a game. The average value fluctuates, as tweets are collected in real-time. If a tweet contains more words than the average word count, it will be considered as long tweet. Otherwise, it will be considered as short tweet. Figure 14 shows the tweet response based on tweet word count. The average and median delay of long tweets is 3 and 10 seconds more than short tweets, respectively. More significantly, the post rate of short tweets is twice that of the long tweets.

In addition to the device, activeness, and tweet length, we also attempted to analyze tweets posted from verified users and users in stadiums. However, due to data unavailability (verified and in-stadium users are less than 0.5% of total users), we could not obtain representative results.

6.2. Filter-Based Event Detection

A key technical contribution is to detect events, such as touchdowns and field goals, from our streams of game-related tweets. We utilize matched filtering to detect game events. To improve the fidelity of the detector, we create a separate template for different groups of Twitter users, and perform the matched filter separately for each group's tweets. We combine the results of all of the matched filters to achieve accurate and timely event detection. We also elaborate on the proper window size and threshold parameters to achieve the best tradeoff between detection delay and true positive rate.

6.2.1. Event Detection with Matched Filtering

Our event detection strives to analyze Twitter streams to detect events. An event occurrence is routinely followed by a patterned volume of event-related tweets as time progresses. We consider tweets that contain a specific event keyword or its variant, e.g., "touchdown" and "TD", as event-related tweets. By aggregating a history of previous events, we can determine the shape of the pattern of tweets that follows an event, i.e., the *event template*. For our event template, we use the tweet response (Figure 11). Our event template accounts for the tweet volume until a certain amount of time has passed after an event. We call this amount of time the *window size* of our event template. To improve the

fidelity of the detector, we create a separate template for the different groups of Twitter users that were explored in Section 6.1.

Once the event templates are established, to perform detection, we adopt *matched filtering*, a technique widely used in signal detection theory to correlate a received signal with a given template. Matched filtering is proven to be the optimal linear filter when dealing with independent additive noise, and is known to perform well in other scenarios as well [70], The matched filter is applied separately for each user group. The matched filtering process is as follows:

Take our event template **V** over the window **W** and take its time-reversed conjugate to form our matched filter **H**.

$$\boldsymbol{H}(t)=\boldsymbol{V} * (\boldsymbol{W} - t)$$

Convolve the current window of the ongoing tweet stream **X** with the matched filter **H**.

$$\boldsymbol{m} = \boldsymbol{X} * \boldsymbol{H}$$

If the resulting signal **m** rises above a threshold, our system detects that an event occurred at that time.

In our dataset, the interval between two adjacent events is always larger than 5 minutes. If **m** remains above the threshold within 5 minutes after a detected event, we consider it is the same event to prevent repetitive reports of the same event.

6.2.2. Addressing User Diversity

As noted in section 6.1, we found that tweets from different user categories had different properties. In particular, mobile and inactive users typically exhibit less of a delay of tweeting about an event than non-mobile and active users. Furthermore, users usually post shorter tweets with event keyword after event happened. Noting these characteristic differences, we form different event templates from their unique tweet responses. Thus, the event templates can be seen in Figure 12, Figure 13, and Figure 14

In order to combine the outputs of the various matched filters, $m_i(t)$, to form the output, $m_{comb}(t)$, we evaluated three prominent combination mechanisms; the Maximum Rule, the Mean Rule, and the Product Rule. The *Maximum Rule* calculates the outcome as the maximum of the matched filters. Formally,

$$m_{max}(t) = \max \{ m_i(t) \}$$

The Mean Rule calculates the outcome as the average of the three matched filters. Formally,

$$m_{mean}(t) = 1/3 \cdot \Sigma\{m_i(t)\}$$

The Product Rule calculates the outcome as the product of the three matched filters. Formally,

$$m_{prod}(t) = \Pi\{m_i(t)\}$$

In addition to these three prominent mechanisms, we also evaluated the combination of inactive users and short tweets, which exhibit shorter delay.

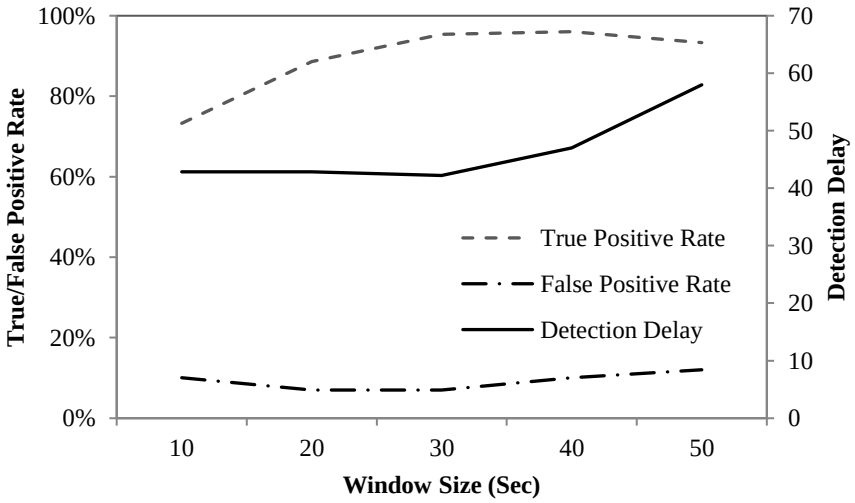


Figure 15: The tradeoff between event detection delay and true positive rate.

$$m_{delay}(t) = 1/2 \cdot (m_{inactive}(t) + m_{short}(t))$$

We compared the performance of these four methods, and found that the Mean Rule performs best. We will illustrate the performance of different combination mechanisms using Receiver Operating Characteristic (RoC) curve in Section 8.3. Indeed, the Mean Rule is known to be especially resilient to noise [71], and useful when inputs are highly correlated. We use $m_{mean}(t)$ in the remainder of the paper, and define $m(t) = m_{mean}(t)$

6.2.3. Parameter Selection

To obtain the optimal window size and threshold, we iterate this matched filter detector method with different window sizes and thresholds through our history of events.

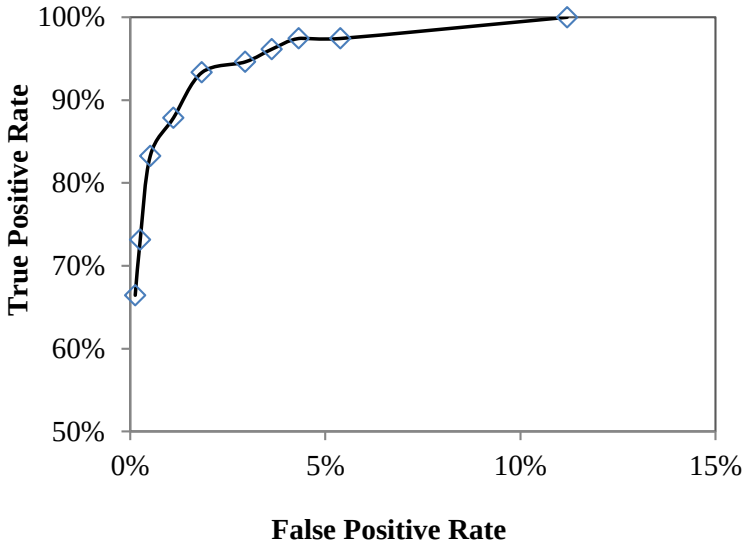


Figure 16: Tradeoff between true positive rate and false positive rate in threshold selection

An appropriate window size should be chosen to reduce detection delay while maintaining high accuracy. A short window will not allow for robust event detection; a lengthy time-series is needed to form a pattern for an event template, and the window should be no shorter than this pattern. However, as the window size increases, the delay of event detection also increases, as we need to sample more tweets before a decision can be made about a particular time. Figure 15 shows the effects of window-size on the accuracy and delay of touchdown detection.

Likewise, the choice of threshold parameter impacts the accuracy of the event detection. A low threshold will recognize events when there are none (false negatives), while a high threshold will fail to recognize events (false positives). The tradeoff in threshold choice is shown in Figure 16. An appropriate threshold must be found to balance this tradeoff to fulfill accuracy requirements.

We rely on historical event data to perform optimal parameter selection. With our Touchdown data, we find the optimal window size to be 30 and threshold to be 8. With these parameters, we achieve 97% true positive rate and less than 4% false positives rate, which confirms the efficacy of our Touchdown event detection. The evaluation results and comparison with threshold based method will be explained in details in Chapter 8.3.

Chapter 7

Regression-Based Event Detection

We have shown that user and tweet characteristics as well as the matched filter method can improve the performance in events detection. In this chapter, we will investigate what features users present when frequent events take place and how to leverage these features for real-time events detection.

7.1. Feature Extraction

The key reasons for feature extraction are to suppress noise and reduce the dimension, i.e. the number of data point, of the original data. Our problem of real-time event detection is closely related to text mining and time series event detection.

In particular, popular features in text mining include frequency count, document clustering, principle component analysis and topic modeling. However, most of these features are not applicable to our work due to our real-time requirement and the noise in

tweets. Firstly, the frequency count provides almost identical result to post rate because the tweet is short. Then, existing clustering and topic modeling methods analyze tens of thousands or more tweets corresponding to tweets in hours to days [32, 58, 72]. Even though, external resources, such as search engine, are used to reduce the noise and improve the results. Given we need to analyze tweets in several seconds, clustering or topic modeling methods cannot provide acceptable results.

Popular features in time series event detection include sampling, sliding window etc. For the same reason as in text mining, some methods are not applicable. For example, many are focused on change points detection. As we discussed in Related Work Section, they can identify the time points at which behavior changes in time series. This can be used to estimate the event time and peak time [30]. However, these methods require the data set in minutes or hours, and predefined number of change points. Moreover, incremental or on-line method is sensitive and will suffer from noise [73].

Then, we explain the rest features that are available to us. Firstly, sampling refers to the process of choosing a subset of data point to be processed. Given limits in data collection and the short, informal tweets, we choose to only analyze tweets that contain event keywords because event keywords express the strongest correlation between tweets and events.

In this section, we explain the features we select. Since we aim to detect the events in real-time, we first discuss the sliding window features that include the most recent tweets. Based on the sliding window approach, we can extract tweet and user

Sliding Window Features:

- Window length (5 sec≤w≤60 sec)
- Window overlap portion (20%, 40%, 60%, 80%)

Features in a Single Window:

- Average tweet length
- Average number of tweets
- Post rate slope, [X(t)-X(t-w)]/w
- Signal to noise ratio
- Standard deviation of tweet timestamps
- Skew of tweet timestamps
- Kurtosis of tweet timestamps
- Number of tweets from mobile devices
- Number of tweets from computer
- Number of tweets from active users
- Number of tweets from influential users (measured by Klout score)

Features in Adjacent Windows:

- Ratio of average tweet length
- Ratio of average number of tweets
- Ratio of post rate slopes
- Ratio to noise ratio
- Ratio of standard deviation of tweet timestamps
- Ratio of skew
- Ratio of kurtosis
- Ratio of the number of tweets from mobile devices
- Ratio of tweets from computer
- Ratio of tweets from active users
- Ratio of tweets from influential users

Figure 17 Summary of extracted features

features either in a single window or between the adjacent windows. The extracted features are summarized in Figure 17.

7.1.1. Sliding Window Features

Sliding Window contains the most recent data points that fulfill the real-time requirement in our problem. In addition, with noisy tweets and rate limit in data collection, feeding data points second by second often cause high false positives. Therefore, aggregating raw data in sliding window become necessary. We use W to denote a window with length w. For example, $X(t)$ with $t_0<t<t_w$ represents post rate in a window with length w from t_0 to t_w. Raw data in sliding window will generate corresponding features.

There are two main features related to sliding window, window length and the window overlap. The selection of window length results in how many tweets we feed in the classifier to detect events. Window length is an important feature because it is closely related to the delay and accuracy. When the window length is longer, more tweets can be analyzed that yields higher accuracy but leads to longer delay. The window overlap shows the portion of the current window overlaps with the previous. The selection of overlap portion determines how many new tweets are considered. The more overlap portion, the fewer new tweets are considered. The current window may contain all new data if the overlap portion is 0. We will show that it is necessary to use overlapped window in next section.

7.1.2. Features in Single Window

Based on the sliding window features, we can extract features in a single window. These features include the properties as discussed in Chapter 5 and Chapter 6, as well as new statistics and user related features.

Recall that we discussed the average number of tweets and the post rate slope in related work [3, 6, 30] and utilized them to detect events in Chapter 5. We keep these two features in this section. Specifically,

$$Average\ number\ of\ tweets = \frac{1}{w}\sum_{t_i-w}^{t_i} X(e_i, t)$$

$$Post\ rate\ slope = \frac{X(e_i, t_i) - X(e_i, t_i - w)}{t_i - (t_i - w)}$$

where *w* is the window length. Post rate slope measures how fast the post rate increases in a single window.

As we have examined tweet properties in matched filter based method, these properties can be inherently utilized in regression based method as features such as the tweet length, tweets from mobile devices and tweets from active users.

$$Short\ Tweets = \frac{1}{w}\sum_{w} X_{short}(e_i, t_i)$$

$$Mobile\ Tweets = \frac{1}{w}\sum_{w} X_{mobile}(e_i, t_i)$$

$$Active\ Tweets = \frac{1}{w}\sum_{w} X_{active}(e_i, t_i)$$

In addition to the tweets from active users, we also take the influence of the user into consideration. Influence describes the reach, impact and expertise of the user on the topic (in this case the football event). We pick Klout scores as the measure of influence due to its status as a de facto standard on Twitter. Klout scores scale from 10 to 100 with 100 indicating the most influential user. The Klout scores are widely available through Klout API with a request limit of maximum 20,000 requests per day. $X_{influence}$ is the number of tweets from influential users whose Klout score is above 60 where the average score is 40 to 50.

$$Influence\ Tweets = \frac{1}{w}\sum_{w} X_{influence}(e_i, t_i)$$

Signal to noise ratio is an extension to post rate. Like any signal detection system, noise also exists in tweet response to events. We consider noise as the tweet including event keywords posted when no event takes place. We measure the level of noise, N by the average post rate of the term for a sufficient long period of time. As a result,

$$SNR = \frac{X(e_i, t)}{N}$$

where $N = \frac{1}{T}\sum_T X(e_i, t)$.

Statistic measures can be considered as features. Standard deviation, skewness and kurtosis, which are second, third and fourth moment of tweet timestamps, respectively, are widely used. We calculate these three features in the following forms:

$$Std = \sqrt{\frac{\sum_1^w (t_i - \bar{t})^2}{w - 1}}$$

$$Skew = \sqrt{\frac{\frac{1}{w}\sum_1^w (t_i - \bar{t})^3}{\left(\frac{1}{w}\sum_1^w (t_i - \bar{t})^2\right)^{3/2}}}$$

$$Kurtosis = \sqrt{\frac{\frac{1}{w}\sum_1^w (t_i - \bar{t})^4}{\left(\frac{1}{w}\sum_1^w (t_i - \bar{t})^2\right)^2}} - 3$$

The standard deviation captures the variation from the average that represents how long a keyword is in use. The skewness captures the asymmetry of the trend in a window that depicts rate of keyword adoption against abandonment. The kurtosis captures the peak shape, or if a keyword is continuously in use over a period of time [10].

Wavelet analysis provides the measurement about when and how the frequency of the signal changes over time. It decomposes a signal into a combination of wavelet coefficient and basis functions. We construct the signal with wavelet analysis according to [32].

7.1.3. Features between Adjacent Windows

Given the sliding window approach, we can design a set of features between the adjacent windows. These features are capable to reveal the relationships between the most recent window and previous windows. Especially, we can leverage the features in a single window and examine the change or trend comparing to those in previous windows.

For each feature in a single window, there will be a counterpart that shows the change between adjacent windows. We use the ratio to measure the difference. Take the number of tweets as an example, the counterpart feature is the ratio between the average number of tweets in current window and the previous window, which is

$$Increasing\ Rate = \frac{\sum_{t_i-w}^{t_i} X(e_i, t)}{\sum_{t_i-2w}^{t_i-w} X(e_i, t)}$$

Figure 17 shows the corresponding features in adjacent windows. We only consider the trend from previous window instead of former windows for two reasons. First, it is the features from most recent data that is caused by the event. Thus, examining the data in the most recent windows is the quickest way to detect event. Second, we have different window lengths as sliding window features. The statistical method will select the best window length that covers enough data for analysis, which will be explained in details in next section.

7.2. Feature Selection

When we extract the features, we use reasoning and experiments to select the best features for our real-time event detection problem. We first select the possible window length and overlap. Then, we select the most correlated features using Pearson correlation coefficient.

Window length is the most important parameter in sliding window because it balances accuracy and delay. While *w* can be any integer that leads to infinite window length features. Testing every possible *w* will cause burden and unnecessary computation.

Now we shrink the length feature set by discussing the following conditions, short window, step size between windows and long window. When window length is short, e.g., $w<5$ second, the number of event tweets in the window is approximately the same as post rate. That is, the post rate is in the range of $0 \leq X(e_i, t) \leq 5$ regardless of event happens. For the same reason, we do not need to increment the window length second by second but set the step size of 5 to 10 seconds according to the window length. For example, after we examine the number of tweets in a window with length $w=40$ second, we do not need to examine $w=41$ because $X(t_1 \text{ to } t_{40}) \approx X(t_1 \text{ to } t_{41})$. On the other hand, when window length is long, e.g., $w>60$ second, the detection delay caused by the long window will be unacceptable for real-time system, which we will show in the Evaluation Section. Thus, the potential window length set is

$$W = \{w | 5 \leq w \leq 60, 5 \leq w_n - w_{n-1} \leq 10\}$$

Window overlap describes how long a window overlap with another when we adopt the sliding window approach. The overlapped part is not necessarily to cover every possibility. On the other hand, the overlapped part should be proportional to the window length. As a result, we will examine the overlap portion with 20%, 40%, 60% and 80%. That is, two windows starts from t_{w1}, t_{w2} with length w, they overlap if $t_{w1} < t_{w2} < t_{w1}+w < t_{w2}+w$ and the overlapped portion is

$$Overlap = \frac{t_{w2} - t_{w1}}{w} = 20\%, 40\%, 60\%, 80\%.$$

Next, we select the most correlated features using Lasso. Specifically, the Lasso estimate $\hat{\beta}^{lasso}$ is defined as the value of β that minimizes

$$\sum_i (y_i - x_i^T \beta)^2 + \lambda \sum_j |\beta_j|$$

where $\lambda \geq 0$ is the complexity parameter that controls the shrinkage. The greater the λ, the greater the amount of shrinkage, i.e. more coefficients shrink all the way to zero [74, 75]. y_i is the binary data point indicates if event happens or not at the moment. As we target at detecting events in seconds, we mark the data point within 1 minute after ground truth as event happens. x_i is a vector where each element is a feature value. For features in adjacent windows, we also take different overlap conditions as separate features in this process.

After the Lasso shrinkage, the result shows that four features are the most relevant to events in real-time:

1. The overlap between adjacent windows is 40%

2. The ratio of average number of tweets between adjacent windows
3. The number of short tweets in a single window
4. The number of tweets from active users in a single window

7.3. Classifier Design

Our event detection strives to analyze Twitter streams to detect events as accurately as possible and as early as possible. To achieve this goal, we need to take following three conditions into consideration. Firstly, the event detection method cannot have high computation complexity in order to reduce delay. Secondly, the method needs to integrate information from multiple features. Thirdly, the method can deal with noisy signal. As a result, we choose regression model because it outperforms more sophisticated methods in situations with low signal-to-noise ratio, small number of training data, or sparse data [74].

We investigate two regression models, linear regression and logistic regression. Both are widely used in classification and describing possible relationships between variables. Especially, logistic regression is used when the outcome is binary result. Specifically, linear regression model has the form

$$y_i = \beta_0 + \beta x_i$$

where y_i is a vector with binary result indicates whether an event happens at ith data point, x_i is the feature vector at ith data point, and β is the parameter vector. Logistic regression model has the form

$$\log\left[\frac{Pr(y_i = 1)}{1 - Pr(y_i = 1)}\right] = \beta_0 + \beta x_i$$

where $Pr(y_i=1)$ indicates the probability of the event happens at *i*th data point, β_0 is the value of the criterion when the predictor is equal to zero, β is the regression coefficient, and x_i is the feature vector at *i*th data point. $Pr(y_i=1)/(1-Pr(y_i=1))$ is also known as the *odds* of event happening.

In the training process, we generate x_i by using the selected features described in Feature Selection Section. We use games with ground truth and follow the standard regression model fitting procedure. Since we adopt the sliding window approach with overlapped adjacent window, the *i*th data point in x_i and y_i indicates the features and results in the window instead of raw data per second.

This regression based detection method slightly outperforms the matched filter based method, which will be demonstrated in details in Chapter 8.3. On the other hand, since this regression based method leverages several features that can potentially detect non-predefined or unanticipated events.

Chapter 8

Implementation and Evaluation

We have implemented SportSense as a real-time web service that visualizes event detection. In this chapter, we present the details of our web service followed by the thorough evaluation of the performance of SportSense. Finally, we present a sample EPG web application and Android application built on SportSense. We present the results based on the 2010-11 and 2011-12 NFL seasons.

8.1. Web Service Realization

We have implemented the SportSense web service in PHP. It consists of the backend for data collection and analysis, and the frontend for web visualization, as illustrated in Figure 18. We host the web service on Amazon EC2, a reliable and reasonably priced cloud computing platform.

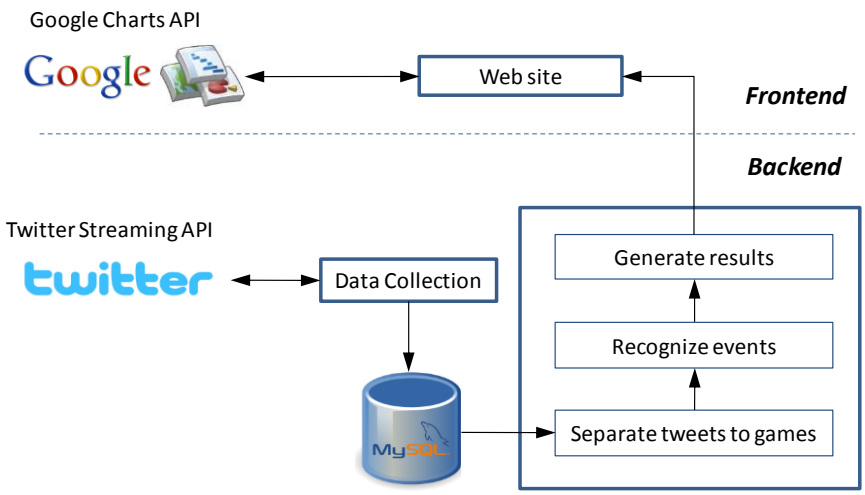
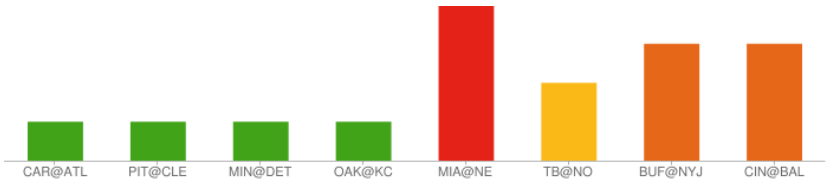


Figure 18: Architecture of the web service implementation

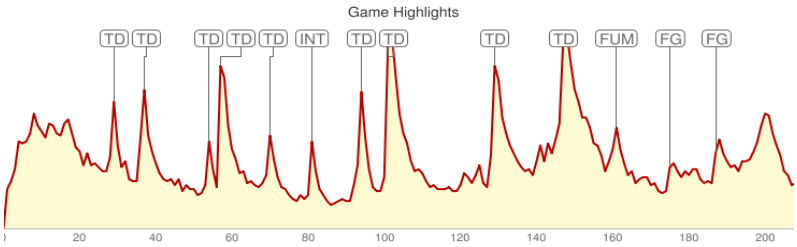
The backend consists of two parallel modules and a MySQL database. The data collection module collects game-related tweets through Twitter Streaming API as described in Section 3.1. Collected tweets are saved in the MySQL database. The event recognition module will retrieve tweets from database, separate tweets to games, recognize events, and generate the results in Google Chart format.

As analyzed above, the backend can introduce many seconds of delay. To minimize this delay, we created multiple threads to maximize the parallelism of data analysis. The data collection module employs one thread to retrieve tweets from Twitter, decode and save the tweets into the MySQL database. The event recognition runs as another thread that retrieves data from the database and analyze them for event recognition.

The frontend visualizes the analysis results using Google Charts API through a website. For ongoing games, the website shows a color-coded bar chart for the “hotness” of all concurrent games according to the post rate of tweets related to each game, as



(a) Color-coded thermometer for concurrent games



(b) Excitement level and event recognition results

Figure 19: Real-time visualization from the website

shown in Figure 19 (a). For each game, the website provides a line chart that draws post rate of tweets related to the game and denotes recognized events, as shown in Figure 19 (b). To update the website in real-time, the website employs an embedded Javascript to pull the results from the backend every two seconds. The line charts and recognized events for past games can be retrieved from the website by either team name or game schedule.

We choose Amazon EC2 to host the web services for its reliability and the full control to the virtual machine that Amazon calls an *instance*. We are using the small (default) standard instance that provides 1.7GB of memory, 1 virtual core and 160GB of local instance storage [76]. This instance is charged on-demand for $0.085 per hour for Linux/Unix. The data transfer cost is $0.1 per GB for data transfer IN and $0-0.08 per GB for data transfer OUT depending on the volume. As our system is always on and generates less than 5GB transfer in data per month, the operational cost is about $60 per month.

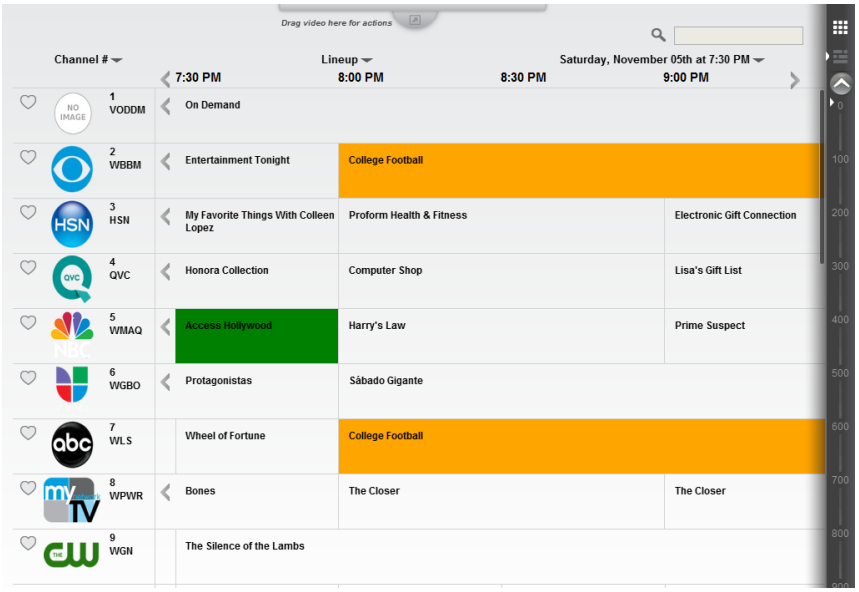


Figure 20: Integration to an Electronic Program Guide (EPG) web application with color coded excitement level indication.

The SportSense website has been running for 2 years monitoring NFL games in 2010-11 and 2011-12 Seasons. The event detection method has also been evolving during this period. The original SportSense utilizes the threshold-based method to detect peaks in tweet volume, which is similar to existing methods in the related work [3, 30]. Different from [3, 30], both threshold-based and matched filter methods can detect events around 40 seconds, which is significantly faster than ESPN website updates (~90 seconds).

8.2. SportSense Powered EPG Application

We have developed the SportEPG application to demonstrate the effectiveness of the SportSense system and its APIs. A primary function of electronic program guides, or EPGs, is to assist users in selecting their desired channel. SportEPG achieves this goal by using the SportSense API to identify and highlight the most tweeted programs in real

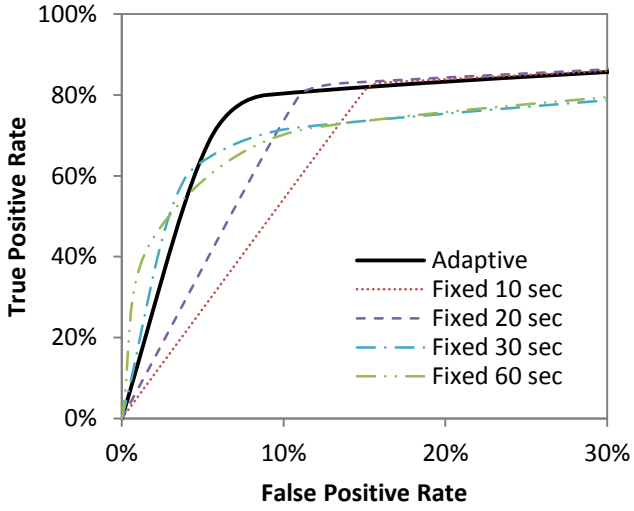


Figure 21: The RoC curves for event detection with different window sizes

time. Thanks to the flexibility of the keyword-based data collection method, we feed TV program names to the Streaming API to support general TV content. Our event detection is still focused on NFL games. As shown in Figure 20, we color the grids to red, orange, yellow and green indicating the "hotness" of the programs. We leverage AJAX (Asynchronous JavaScript And XML) methods to retrieve analysis results from the SportSense server asynchronously, without interfering with the existing page.

One of the benefits of a real-time updating EPG is that it is accessed frequently and presumably in most cases on a companion device such as a smartphone or a tablet. In turn, the frequent, constant attention to the EPG can be exploited through relevant advertising.

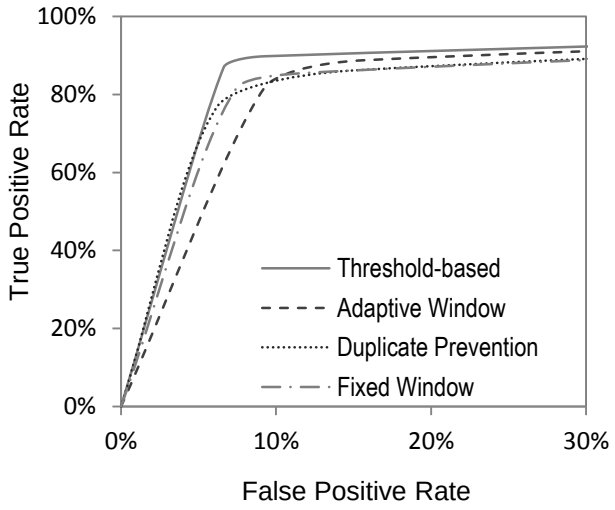


Figure 22: The RoC curves for events detection with improvement methods described in Chapter 5

8.3. Evaluation

Next, we will show the accuracy in major events detection. We evaluate the SportSense designs using broadcasted NFL games. The evaluation proves that both threshold-based and matched filter methods can detect events around 40 seconds while achieve more than 90% true positives, less than 10% false positives.

8.3.1. Threshold-based Detection Performance

Like any binary classifier, the detection stage can make two types of errors: reporting an event when nothing happens (False positive) and reporting nothing when an event happens (False Rejection). Using all 435 events, 163 touchdowns, 68 interceptions, 91 field goal, and 113 fumbles, happened in 33 games in the Week 16, 17 and playoffs of the 2010-2011 NFL season, we illustrate the effectiveness of our threshold-based method.

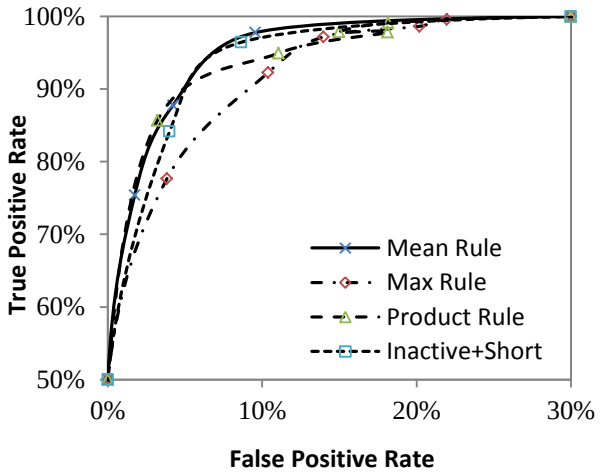


Figure 23: The RoC curves for different combination mechanisms of matched filter detectors. The mean rule outperforms the others.

Figure 21 shows the Receiver Operating Characteristics (RoC) Curves for our adaptive methods and those based on a fixed sliding window. We compared the adaptive window approach with the fixed window approaches. From the results shown in the RoC curve, the adaptive window size outperforms the fixed window size.

Then we show that the adaptive window and duplicate prevention techniques can effectively improve the event detection accuracy.

Figure 22 shows the Receiver Operating Characteristics (RoC) Curves for our basic solution and those with improvement techniques. From the results shown in the RoC curve, the adaptive window size introduced more false positive rate than the fixed window size because the shorter window size leads to duplicate event alerts as described in Chapter 5. Based on the basic solution, the duplicate alerts prevention does not improve a lot on the false positive rate. However, after applying all the techniques, the true positive rate reaches nearly 90% and the false positive rate is only 8%.

8.3.2. Filter-based Detection Performance

To evaluate the effectiveness of the matched filter detector of SportSense, we use leave-one-out cross-validation (LOOCV) on the 18 games in 2011-12 Season. LOOCV uses a single game as the validation data, and the remaining games as the training data. There are in total 100 touchdowns, an average of 5 or 6 touchdowns per game. We show that the combination of matched filter detector of different user types can achieve 98% true positive rate and 9% false positive rate.

Figure 23 shows the Receiver Operating Characteristics (RoC) Curves for our single matched filter detector and the combination of matched filter detectors for different user groups. The single matched filter detector approach achieves 96% true positive rate and has 13% false positive rate. On the other hand, the combination matched filter detectors performs better, achieving 98% true positive rate and has 9% false positive rate. Figure 24 shows the performance of the threshold based method and matched filter method for touchdowns and field goals. The threshold-based method performs significantly poorer than the matched filter methods.

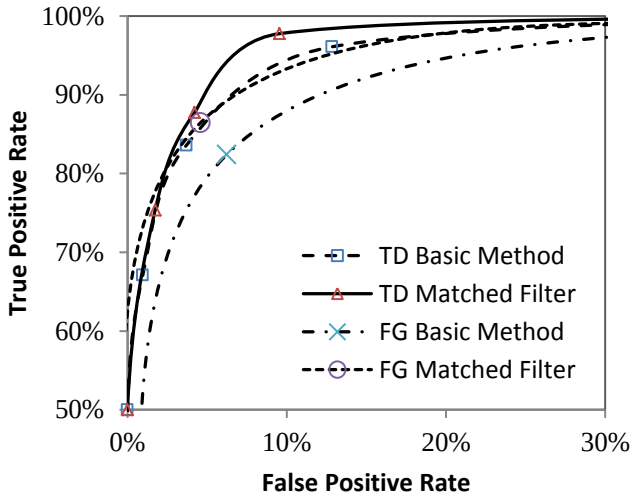


Figure 24: RoC curves for basic temperature based and matched filter detectors. Matched filter performs better for touchdowns and field goals.

We also examine the delay of our detection, which is introduced from two parts, our system and the detection stage. With the parallelism design in data collection and data analysis, the delay from the detection stage includes computation delay and the delay due to the matched filter window size. The computation as introduced in Section 6.2.1 is mainly a convolution operation, which is very fast. Figure 25 shows the delay of events detected. All three methods have an average delay around 45 seconds. The median value of the threshold-based method is 5 seconds shorter than the other two methods but the threshold-based method detects fewer events due to the tradeoff between accuracy and window size. Interestingly, the combination of inactive and short tweets does not outperform the matched filter method. Both have similar detection delay distribution. For the purpose of comparison, we note that EPSN takes about 90 seconds to update their webpage. SportSense detects 60% of events within 40 seconds mainly because of the window size. All detected events have delay less than 90 seconds. As a result, SportSense can update results faster than ESPN updates their web page.

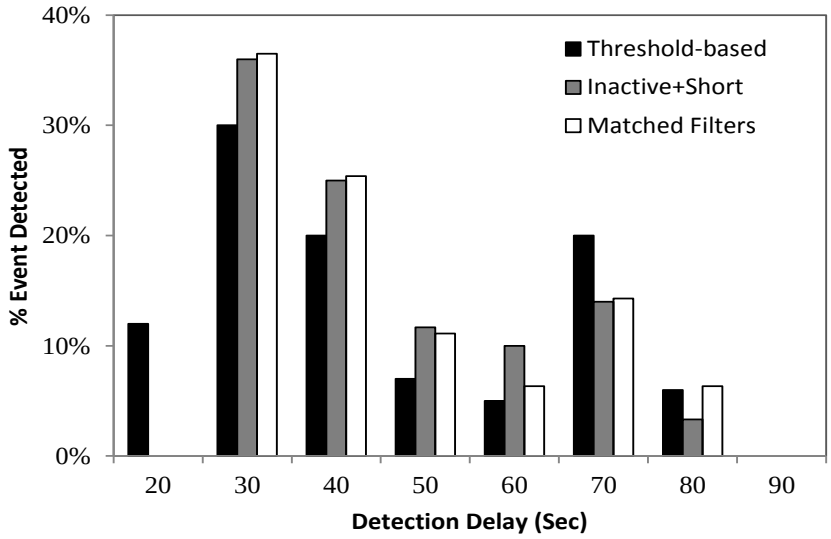


Figure 25: The distribution of event detection delay.

8.3.3. Regression-based Detection Performance

To evaluate the regression based detector, we use the same leave-one-out cross-validation (LOOCV) on the 18 games in 2011-12 Season as we used for matched filter based method. The results show that regression based method improves the performance to achieve shorter latency and fewer false negatives as shown in Figure 26 and Figure 27.

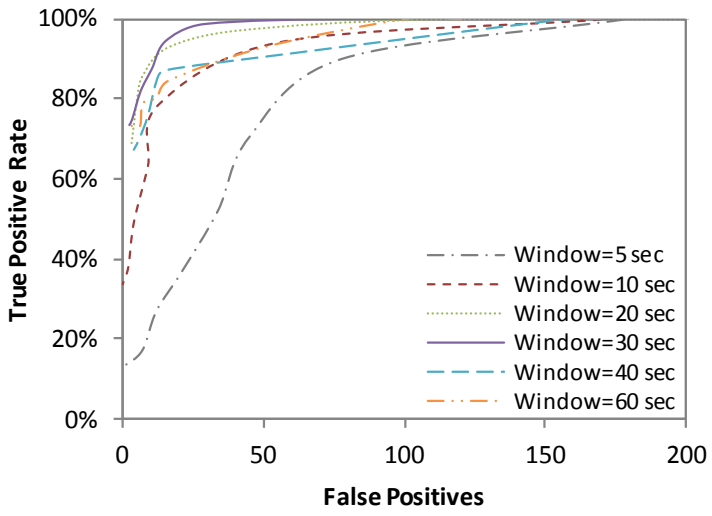


Figure 26 The RoC curves for event detection with different window sizes

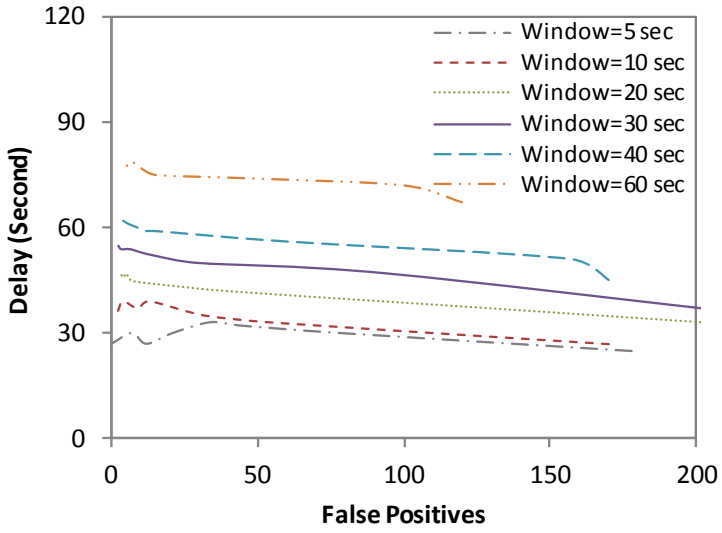


Figure 27 The AMOC curves for event detection with different window sizes

Figure 26 shows the Receiver Operating Characteristics (ROC) Curves for the linear regression method with different window lengths. In the figure, the Y axis represents the true positive rate, the X axis represents the number of false positives rather than false positive rate because there is no event happens in majority of data points, the false positive rate is always close to zero. The ROC curves demonstrate the trade-off

between window length and false positives. When the window length is extremely short, i.e. 5 seconds, the false positives are significantly more than longer windows. This is due to the insufficient information in the short window so that noises are considered as events. The accuracy is the best when window length is between 20 to 30 seconds and the regression approach achieves 95% true positive rate and has 14 false positives.

Figure 27 shows the Activity Monitoring Operating Characteristic (AMOC) curves for our regression method with different window lengths. The Y axis indicates the detection delay; the X axis indicates the number of false positives and false negative rate, respectively. The AMOC curves present the trade-off between window length and delay. The figures show that the delay spans from 20 to 80 seconds. The shorter window length, the shorter delay can be achieved. Note the delay shown in the figure is the delay from event happens to our system detects the event that includes three part, i.e., human delay, Twitter delay and system delay according to our discussion in Chapter 4. As a result of Figure 26 and Figure 27, the regression based method performs the when the window length is 20 seconds.

8.4. Discussion and Summary

Next, we discuss and summarize the advantages and disadvantages of these three methods, i.e., threshold, matched filter and regression based methods. From the evaluation, we have demonstrated that all three methods are able to accurately detect major events with more than 90 % accuracy. More importantly, 60% of events can be

detected within 40 second and all events can be detected within 90 second. In addition to the performance, these three methods have its unique property, which will be examined in this section.

We start from the threshold based method which has many advantages including flexibility, easy to implement and compute. The threshold based method is the most flexible method as it requires little training data comparing to matched filter and regression based methods where flexibility indicates the capability of detecting a new set of events. To detect a set of new events, threshold based method simply needs to update the lexicon of the targeted events and a threshold. To select the best lexicon and threshold, it requires the domain knowledge, which is usually widely available. For example, if we would like to monitor a different type of game, such as soccer or baseball, instead of American football, the lexicon can be easily found from professional sports websites. Even in a broader domain, e.g., TV shows or breaking news, the related keywords can be retrieved from the TV program schedule, introduction, cast information and news websites. The threshold selection is not complicated either. The threshold is not necessarily to be a certain value. Instead, events can be successfully detected within a range of threshold. However, the threshold based method has two major limitations. First, it is neither the fastest method nor the most accurate according to our evaluation results. Second, the threshold based method heavily relies on the pre-defined lexicon and the increase of the post rate. It does not take other features into consideration. As a result, if the event does not convey enough information in the post rate of the predefined keyword, the performance will be severely affected.

The advantage of the matched filter based method is primarily the detection performance and robustness because it considers more features from tweet text and user profile. From the evaluation results, the matched filter based method outperforms the threshold based method in terms of accuracy and latency, and performs about the same as the regression based method. An important reason of the improvement is the additional properties and features. These features capture the event from various properties of the tweets, i.e., tweet length, devices, and user activeness, when events take place. Although the matched filter based method requires training, the event templates are applicable to various events in different domain as most online events share a limited set of patterns [12]. On the other hand, matched filter based method is not as flexible as the threshold based method. Moreover, matched filter based method requires the comparison between data and template for each feature, which is not efficient in real-time system. There also exist potential features to be considered as we discussed in Chapter 7.

The regression based method performs the best but it is only slightly better than the matched filter based method. The regression based method examines a more complete set of features that include three categories, sliding window, tweet text and user profile. We use statistical subset shrinkage method, Lasso, to select the most relevant features and we use linear regression as the classifier to detect events. The extracted features are common in many events so that this method not only performs well for detecting predefined events but also applicable for discovering new events.

Chapter 9

Future Work

In previous chapters, we obtained the knowledge of Twitter APIs, delay characteristics and events features. We demonstrated the capability of extracting topic related tweets and detecting predefined events in real-time. Based on our SportSense platform, our future work includes two major directions including unanticipated / non-predefined events recognition and weak events detection.

9.1. Unanticipated Events Recognition

At this moment, our work is limited to events for which keywords can be predetermined, such as for NFL games. What would be more useful and challenging is to recognize events that are not anticipated, and therefore do not have predefined keywords. Leveraging the work reported here, we are actively pursuing this goal. That is, based on the features we extracted in regression based detector, we can dynamically determine the keyword or tweet that contains the same feature. For example, we may select high quality

users using measurement matrix derived from the user characteristic features, or existing measurement matrix, such as Klout. Then, we can dynamically extract keyword or tweet posted by high quality users. As a result, this work can be easily applied to other domains such as real-time incident/disaster management.

9.2. Weak Events Detection

In addition to strong events, weak events detection is our next goal. This work is focused on strong events, which are well-known, major and moderately frequent events, such as a touchdown in American Football or a goal in soccer. Weak events, such as a spectacular sack in American Football or a give-and-go in soccer, are also of high interest to fans even though these events do not contribute to the final score. Given that strong events have the weak signal when they just take place, weak events detection can improve the strong events detection speed. Next, we will discuss the potential direction of extending current work to weak events detection.

First of all, we define the weak events and distinguish it from strong event using the increasing of post rate and frequency of the events. In Chapter 7, an important feature of strong event is the ratio of post rates between windows or simply the increasing of post rate. Many also use increasing of post rate to label or detect events in literature [3, 6, 30, 36]. Therefore, we use increasing of post rate to distinguish weak events from strong events. That is, the peak is not recognizable for weak event using the methods in literature. While for strong events, one or more peaks can be recognized in a given series of post rate of event keyword or tweet. In addition, the frequency of weak events should

be in the same magnitude of strong events, which is the number of peaks in a given series data.

To achieve weak events detection, we need to further investigate the features in the tweet text and the user's profile. For example, in our preliminary experiments on user's profile, we found that the influential users who receive higher Klout score often post tweets about weak events more accurately, but not necessarily faster [77]. We will continue in this direction and strive to achieve weak events detection in real-time.

Chapter 10

Conclusion

This thesis introduced the design and implementation of SportSense system, the first of its kind that could detect moderately frequent events in real-time. We investigated into the delay characteristics of users, Twitter, our system and we pushed the limit of Twitter as a real-time human sensor. We studied the Twitter APIs and found that the Twitter API limits can be overcome using proper scoping and event detection keywords. We collected hundreds of millions of game related tweets. We demonstrated that moderately frequent and diverse social and physical events like those of NFL games can be reliably recognized within 40 seconds using tweets properly collected in real-time. Our results also suggest that the threshold, matched filter, and regression based methods are effective in Twitter event detection. We hope this new capability will inspire more applications in the pervasive computing community.

References

[1] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in *ACM WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, San Jose, California, 2007, pp. 56-65.

[2] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi, "Measuring user influence in Twitter: The million follower fallacy," in *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2010.

[3] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in *ACM World Wide Web (WWW)*, Raleigh, North Carolina, USA, 2010, pp. 851-860.

[4] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "TwitterStand: news in tweets," in *ACM SIGSPATIAL Advances in Geographic Information Systems*, Seattle, Washington, 2009, pp. 42-51.

[5] D. A. Shamma, L. Kennedy, and E. F. Churchill, "Tweet the debates: understanding community annotation of uncollected sources," in *ACM SIGMM workshop on Social Media*, Beijing, China, 2009, pp. 3-10.

[6] S. Zhao, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "Human as Real-Time Sensors of Social and Physical Events: A Case Study of Twitter and Sports Games," *Arxiv preprint arXiv:1106.4300*, 2011.

[7] S. Zhao, L. Zhong, J. Wickramasuriya, V. Vasudevan, R. LiKamWa, and A. Rahmati, "SportSense: Real-Time Detection of NFL Game Events from Twitter," *arXiv preprint arXiv:1205.3212*, 2012.

[8] A. Kazeniac. (2009). *Social Networks: Facebook Takes Over Top Spot, Twitter Climbs*. Available: http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network/

[9] K. Rosman. (2011). *Eat Your Vegetables, and Don't Forget to Tweet*. Available: http://finance.yahoo.com/news/pf_article_112952.html

[10] J. Huang, K. M. Thornton, and E. N. Efthimiadis, "Conversational tagging in twitter," in *ACM Hypertext and Hypermedia*, Toronto, Ontario, Canada, 2010, pp. 173-178.

[11] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?," in *ACM World Wide Web (WWW)*, Raleigh, North Carolina, USA, 2010, pp. 591-600.

[12] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *ACM international conference on Web Search and Data Mining (WSDM)*, Hong Kong, China, 2011, pp. 177-186.

[13] P. Andre, M. Bernstein, and K. Luther, "Who gives a tweet?: evaluating microblog content value," in *ACM conference on Computer Supported Cooperative Work (CSCW)*, Seattle, Washington, USA, 2012, pp. 471-474.

[14] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi, "Short and tweet: experiments on recommending content from information streams," in *ACM*

Conference on Human Factors in Computing Systems (CHI), Atlanta, Georgia, USA, 2010, pp. 1185-1194.

[15] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on Twitter: human, bot, or cyborg?," in *ACM Annual Computer Security Applications Conference (ACSAC)*, Austin, Texas, 2010, pp. 21-30.

[16] M. Naaman, J. Boase, and C.-H. Lai, "Is it really about me?: message content in social awareness streams," in *ACM Computer Supported Cooperative Work (CSCW)*, Savannah, Georgia, USA, 2010, pp. 189-192.

[17] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "TwitterRank: finding topic-sensitive influential twitterers," in *ACM international conference on Web Search and Data Mining (WSDM)*, New York, New York, USA, 2010, pp. 261-270.

[18] K. R. Canini, B. Suh, and P. L. Pirolli, "Finding Credible Information Sources in Social Networks Based on Content and Social Structure," in *IEEE Privacy, Security, Risk and Trust (PASSAT), IEEE international conference on Social Computing (SocialCom)*, 2011, pp. 1-8.

[19] A. Pal and S. Counts, "Identifying topical authorities in microblogs," in *ACM international conference on Web Search and Data Mining (WSDM)*, Hong Kong, China, 2011, pp. 45-54.

[20] B. Hecht, L. Hong, B. Suh, and E. H. Chi, "Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles," in *ACM Conference on Human Factors in Computing Systems (CHI)*, Vancouver, BC, Canada, 2011, pp. 237-246.

[21] K. D. Rosa, R. Shah, B. Lin, A. Gershman, and R. Frederking, "Topical Clustering of Tweets," in *ACM Social Web Search and Mining (SWSM)*, 2011.

[22] X. Changsheng, Z. Yi-Fan, Z. Guangyu, R. Yong, L. Hanqing, and H. Qingming, "Using Webcast Text for Semantic Event Detection in Broadcast Sports Video," *Multimedia, IEEE Transactions on*, vol. 10, pp. 1342-1355, 2008.

[23] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen, "Microblogging during two natural hazards events: what twitter may contribute to situational awareness," in *ACM Human Factors in Computing Systems (CHI)*, Atlanta, Georgia, USA, 2010, pp. 1079-1088.

[24] J. Hannon, K. McCarthy, J. Lynch, and B. Smyth, "Personalized and automatic social summarization of events in video," in *ACM Intelligent User Interfaces (IUI)*, Palo Alto, CA, USA, 2011, pp. 335-338.

[25] D. Chakrabarti and K. Punera, "Event summarization using tweets," in *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2011.

[26] S. A. Golder and M. W. Macy, "Diurnal and Seasonal Mood Vary with Work, Sleep, and Daylength Across Diverse Cultures," *Science*, vol. 333, pp. 1878-1881, September 30, 2011 2011.

[27] J. Bollen, A. Pepe, and H. Mao, "Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena," in *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[28] N. A. Diakopoulos and D. A. Shamma, "Characterizing debate performance via aggregated twitter sentiment," in *ACM Human Factors in Computing Systems (CHI)*, Atlanta, Georgia, USA, 2010, pp. 1195-1198.

[29] P. H. C. Guerra, A. Veloso, J. Wagner Meira, and V. Almeida, "From bias to opinion: a transfer-learning approach to real-time sentiment analysis," in *ACM Knowledge Discovery and Data mining (SIGKDD)*, San Diego, California, USA, 2011, pp. 150-158.

[30] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *ACM Human Factors in Computing Systems (CHI)*, Vancouver, BC, Canada, 2011, pp. 227-236.

[31] S. Zhao, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "Analyzing twitter for social tv: Sentiment extraction for sports," in *Proceedings of the 2nd International Workshop on Future of Television*, 2011.

[32] J. Weng and B. S. Lee, "Event detection in twitter," in *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2011.

[33] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM'11)*, 2011.

[34] A. J. Quinn and B. B. Bederson, "A taxonomy of distributed human computation," 2009.

[35] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger, "Crowds in two seconds: enabling realtime crowd-powered interfaces," in *ACM Symposium on User Interface Software and Technology (UIST)*, Santa Barbara, California, USA, 2011, pp. 33-42.

[36] Y. Qu, C. Huang, P. Zhang, and J. Zhang, "Microblogging after a major disaster in China: a case study of the 2010 Yushu earthquake," in *ACM Computer Supported Cooperative Work (CSCW)*, Hangzhou, China, 2011, pp. 25-34.

[37] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "TurKit: human computation algorithms on mechanical turk," in *ACM Symposium on User Interface Software and Technology (UIST)*, New York, New York, USA, 2010, pp. 57-66.

[38] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," presented at the Proceedings of the 1994 ACM SIGMOD international conference on Management of data, Minneapolis, Minnesota, USA, 1994.

[39] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Foundations of Data Organization and Algorithms*. vol. 730, D. Lomet, Ed., ed: Springer Berlin Heidelberg, 1993, pp. 69-84.

[40] M. Yang-Sae, W. Kyu-Young, and L. Woong-Kee, "Duality-based subsequence matching in time-series databases," in *Data Engineering, 2001. Proceedings. 17th International Conference on*, 2001, pp. 263-272.

[41] Y.-S. Moon, K.-Y. Whang, and W.-S. Han, "General match: a subsequence matching method in time-series databases based on generalized windows," presented at the Proceedings of the 2002 ACM SIGMOD international conference on Management of data, Madison, Wisconsin, 2002.

[42] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," presented at the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C., 2003.

[43] E. Keogh, S. Lonardi, and B. Y.-c. Chiu, "Finding surprising patterns in a time series database in linear time and space," presented at the Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, 2002.

[44] E. Keogh, J. Lin, and A. Fu, "HOT SAX: efficiently finding the most unusual time series subsequence," in *Data Mining, Fifth IEEE International Conference on*, 2005, p. 8 pp.

[45] P. K. Chan and M. V. Mahoney, "Modeling multiple time series for anomaly detection," in *Data Mining, Fifth IEEE International Conference on*, 2005, p. 8 pp.

[46] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *Image Processing, IEEE Transactions on*, vol. 12, pp. 796-807, 2003.

[47] Y. Rui, A. Gupta, and A. Acero, "Automatically extracting highlights for TV Baseball programs," in *ACM Multimedia*, Marina del Rey, California, United States, 2000, pp. 105-115.

[48] D. Zhang and S.-F. Chang, "Event detection in baseball video using superimposed caption recognition," in *ACM Multimedia*, Juan-les-Pins, France, 2002, pp. 315-318.

[49] L. He, E. Sanocki, A. Gupta, and J. Grudin, "Auto-summarization of audio-video presentations," in *ACM Multimedia*, Orlando, Florida, United States, 1999, pp. 489-498.

[50] S. Nepal, U. Srinivasan, and G. Reynolds, "Automatic detection of 'Goal' segments in basketball videos," in *ACM Multimedia*, Ottawa, Canada, 2001, pp. 261-269.

[51] J. Wang, C. Xu, E. Chng, L. Duan, K. Wan, and Q. Tian, "Automatic generation of personalized music sports video," in *ACM Multimedia*, Hilton, Singapore, 2005, pp. 735-744.

[52] K. Petridis, S. Bloehdorn, C. Saathoff, N. Simou, S. Dasiopoulou, V. Tzouvaras, S. Handschuh, Y. Avrithis, Y. Kompatsiaris, and S. Staab, "Knowledge representation and semantic annotation of multimedia content," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 153, pp. 255-262, 2006.

[53] C.-P. Wei and Y.-H. Lee, "Event detection from online news documents for supporting environmental scanning," *Decision Support Systems*, vol. 36, pp. 385-401, 2004.

[54] Y. Yang, J. Zhang, J. Carbonell, and C. Jin, "Topic-conditioned novelty detection," in *ACM international conference on Knowledge Discovery and Data Mining (SIGKDD)*, Edmonton, Alberta, Canada, 2002, pp. 688-693.

[55] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, "Parameter free bursty events detection in text streams," in *International conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005, pp. 181-192.

[56] G. Kumaran and J. Allan, "Text classification and named entities for new event detection," in *ACM conference on Research and Development in Information Retrieval (SIGIR)*, Sheffield, United Kingdom, 2004, pp. 297-304.

[57] Z. Li, B. Wang, M. Li, and W.-Y. Ma, "A probabilistic model for retrospective news event detection," in *ACM conference on Research and Development in Information Retrieval (SIGIR)*, Salvador, Brazil, 2005, pp. 106-113.

[58] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to Twitter," in *Human Language Technologies (HLT) Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Los Angeles, California, 2010, pp. 181-189.

[59] Twitter API Announcements, "Update on Whitelisting, http://groups.google.com/group/twitter-api-announce/browse_thread/thread/1acd954f8a04fa84/9321c609cd8f7751?lnk=gst&q=whitelist#9321c609cd8f7751," 2011.

[60] Borasky Research Journal. (2010). *The Twitter Streaming API -- How It Works and Why It's A Big Deal, http://borasky-research.net/2010/01/06/the-twitter-streaming-api-how-it-works-and-why-its-a-big-deal/*. Available: http://borasky-research.net/2010/01/06/the-twitter-streaming-api-how-it-works-and-why-its-a-big-deal/

[61] Twitter. (2010). *Search API: High-Volume and Repeated Queries Should Migrate to Streaming API*. Available: http://groups.google.com/group/twitter-api-announce/browse_thread/thread/c8c713bb63fac24c?pli=1

[62] Twitter blog. (2011). *#superbowl, http://blog.twitter.com/2011/02/superbowl.html*.

[63] Z. Wang, X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones," in *Proc. ACM HotMobile*, 2011.

[64] "Twitter Development Talk, http://groups.google.com/group/twitter-development-talk."

[65] Twitter Development Talk, "Rate limit for streaming API, http://groups.google.com/group/twitter-development-talk," 2010.

[66] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proc. ACM WWW*, 2010.

[67] Y. Qu, C. Huang, P. Zhang, and J. Zhang, "Microblogging after a major disaster in China: a case study of the 2010 Yushu earthquake," in *ACM conference on Computer Supported Cooperative Work (CSCW)*.

[68] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "TwitterStand: news in tweets," in *Proc. ACM SIGSPATIAL*, 2009.

[69] Z. Wang, X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones?," in *HotMobile*, 2011.

[70] R. D. Hippenstiel, *Detection theory: applications and digital signal processing*: CRC, 2002.

[71] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On Combining Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 226-239, 1998.

[72] M. S. Bernstein, B. Suh, L. Hong, J. Chen, S. Kairam, and E. H. Chi, "Eddi: interactive topic-based browsing of social status streams," presented at the Proceedings of the 23nd annual ACM symposium on User interface software and technology, New York, New York, USA, 2010.

[73] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence,* vol. 24, pp. 164-181, 2011.

[74] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer,* vol. 27, pp. 83-85, 2005/06/01 2005.

[75] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences,* vol. 2, pp. 183-202, 2009.

[76] Amazon. *Amazon EC2*, *http://aws.amazon.com/ec2/*. Available: http://aws.amazon.com/ec2/

[77] V. Vasudevan, J. Wickramasuriya, S. Zhao, and L. Zhong, "Is Twitter a Good Enough Social Sensor for Sports TV?."